UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

FACULTAD DE INGENIERÍAS

Título: Sistema autónomo para control de préstamo de equipos. Caso de estudio: Consultorio Odontológico de la Espriella

Autor: Melet David Chirino Caicedo

Jurado

Jurado

Director:

Cartagena, June 30, 2017

Sistema autónomo para control de préstamo de equipos. Caso de estudio: Consultorio Odontológico De la Espriella

Melet David Chirino Caicedo
Director: Juan Carlos Martinez Santos

Universidad Tecnológica de Bolívar
Facultad de Ingenierías
Programa de Ingeniería Mecatrónica
Cartagena

June 30, 2017

Sistema autónomo para control de préstamo de equipos. Caso de estudio: Consultorio Odontológico De la Espriella

Melet David Chirino Caicedo

Trabajo de grado para optar al título de

Ingeniero Mecatrónico

Director:
Juan Carlos Martínez Santos

Universidad Tecnológica de Bolívar
Facultad de Ingenierías
Cartagena

June 30, 2017

Resumen

La carrera de odontología es una carrera popular en Cartagena, pero es una de las carreras mas costosas debido a la cantidad de herramientas que requiere y la complejidad del equipo que se necesita usar. Actualmente la universidad proporciona la mayoría de los equipos necesarios pero estan sujetos a horarios muy estrictos para que los puedan usar todos los estudiantes y si el estudiante no termina sus de atender a sus pacientes dentro del horario establecido no podrá seguir usando el dispositivo. Esta disponibilidad reducida lleva a estudiantes, principalmente estudiantes de postgrado, a buscar otros sitios para atender a sus pacientes. En cartagena hay consultorios odontológicos creados para solucionar este problema. Estos consultorios alquilan sus equipos por horas para que los estudiantes realicen sus operaciones dentales que normalmente son de larga duración y alta complejidad. Inclusive es normal que estudiantes de pregrado alquilen el consultorio para recibir clases de un tutor sobre operaciones dentales. Uno de estos consultorios esta teniendo problemas económicos debido al poco control ejercido dentro de este, sin embargo, este control puede ser facilmente implementado usando los conocimientos sobre tecnología adquiridos en la carrera de Ingenieŕia Mecatrónica. Este trabajo consiste en desarrollar un sistema para manipular las variables necesarias para aumentar la productividad del consultorio odontológico.

Agradecimientos

Le agradezco a Dios por haberme acompañado en este camino y haberme dado las facultades físicas e intelectuales para llegar hasta aquí. Por haberme dado paciencia y fortaleza en los momentos difíciles y sobretodo una vida llena de experiencias y felicidad.

Agradezco a mi madre Cristina que siempre fue un gran ejemplo de superación dando todo por sus hijos, logrando cada una de las metas que se ha propuesto y sacrificando siempre todo por hacerme una persona grande en mente y cuerpo.

A mi padre Ivan por todos sus consejos y apoyo en los momentos difíciles, por prepararme siempre para la vida con sus conocimientos avanzados y por demostrarme todos los días que con amor y dedicación todo se puede.

A mis hermanos Alex Paul y Luis que siempre ha sido mi compañía inseparable, uno demostrandome que con paciencia se pueden lograr muchas cosas y el otro recordandome la inocencia que vive dentro de mi que siempre me saca de situaciones difíciles.

Una mención a mi prima Carmen que estuvo allí compartiendo cada etapa de mi vida, nuestra abuela estuviera orgulloso de los dos. Y gracias por Martina que me ha recordado que el amor existe.

Gracias también a mi gran familia que siempre estuvieron allí de una u otra forma enseñándome como se puede salir desde lo mas bajo con trabajo y esfuerzo.

Le agradezco su confianza al odontólogo Julio De la Espriella y a sus hijas Juliana,

Silvana y la pequeña Marianita, gracias por haberme confiado su mayor posesión para modificarla y hacerla mas productiva.

Le agradezco a mi tutor Juan Carlos Martínez Santos por haberme ayudado en este proceso creyendo en mi desde muy temprano, que me ha tenido paciencia y ha compartido sus conocimientos siempre para poder llegar a este momento.

Le agradezco la confianza, apoyo y dedicación a mis profesores: Andrés Marrugo, Sonia Contreras, Eugenio Yime y Enrique Gonzales. Por haber compartido sus conocimientos y sobretodo su amistad.

A todos mis amigos que estuvieron conmigo de manera directa o indirecta en esta etapa de mi vida. Gracias por tantas vivencias y momentos que jamás olvidaré. Una mención especial mi gran amigo Raúl a quien admiro mucho y felicito por sus logros, te lo mereces.

A todas aquellas personas que ya no están conmigo que me ayudaron en mi crecimiento personal y espiritual, destacando a mi abuelita Carmen una de las personas mas influyentes e importantes de mi existencia. Abuelita siempre la llevaré en mi corazón.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1.
Introducción

En Cartagena la odontología es una carrera muy popular por lo que no es extraño encontrar muy frecuentemente una persona que la halla estudiado o que la esté estudiando. Esto gracias a que hay una universidad pública que tiene una facultad propia de esta carrera. La odontología es una carrera costosa debido a la cantidad de herramientas que los profesionales utilizan para realizar sus labores. Afortunadamente, la mayoría de las herramientas necesarias son proporcionadas por la universidad, pero al ser públicas estas están sujetas a horarios poco flexibles y si los estudiantes no terminan de atender a los pacientes dentro de este horario deben buscar otro consultorio donde trabajar. Esto ocurre mucho con estudiantes de postgrado quienes realizan operaciones de larga duración y no logran a terminarlas en la universidad. Para poder realizar estas operaciones fuera de la universidad los estudiantes deben recurrir a alquilar un consultorio odontológico donde pagan por el tiempo que necesitan para terminar sus operaciones. En la Figura 1.1 se muestra un consultorio odontológico regular y en la Figura 1.1 Se puede observar una odontóloga realizando una operación dental.

Uno de estos consultorios para alquilar es muy usado en horas de la tarde, noche

y los fines de semana por profesionales de la carrera, estudiantes de maestría e incluso estudiantes de pregrado que le pagan a tutores para que les enseñen a realizar procedimientos de complejidad media.

Figura 1.1: Consultorio regular de odontología.

Figura 1.2: Estudiante de postgrado odontología realizando una operación dental.

El consultorio citado últimamente está teniedo problemas económicos por falta

de control sobre sus equipos al momento de alquilarlos dado que una gran parte de sus clientes se aprovechan de la privacidad brindada para atender a sus pacientes y exceden los límites de uso del servicio. Del mismo modo, las ambigüedades al momento de alquilar los dispositivos como la hora exacta del inicio, el uso después de finalizado el tiempo, entre otros.

Estos problemas se crean por el hecho de que no se puede controlar el servicio de manera remota sin invadir la privacidad del cliente, además el método de vigilancia usado era ineficiente y la única manera de limitarles el servicio a los clientes era desconectar una manguera que lleva aire comprimido al consultorio, pero muchos aún así seguían usando el dispositivo.

Hoy en día la tecnología permite controlar variables varias para dar soluciones a problemas parecidos. Hay ejemplos como en parques temáticos lso clientes recargan una tarjeta y al entrar a una atraccion en particular esta le habilita el acceso y le descuenta el dinero gastado. Uno de estos sistemas puede ser fácilmente adaptables para controlar los dispositivos del consultorio y así evitar que los usuarios usen los dispositivos alquilados por mas tiempo del permitido.

Uno de los sistemas mas prácticos y económicos utilizados en la actualidad para estos casos son los sistemas de tags NFC que son muy populares en sistemas de transporte masivo principalmente porque manejan muy bien variables numéricas sin necesidad inicial de manejar una base de datos lo que evita que los dispositivos tengan que tener una comunicación permanente a una base de datos.

Además de controlar los tiempos de uso, permiten tener registros de bastante utilidad dentro del sistema para su analisis posterior como horas de uso, dinero recaudado, horas de mayor frecuencia, actividad de los usuarios, entre otras. Finalmente estos tags NFC brindan una alta seguridad y hacen que el sistema sea difícil de quebrar.

1.1 Objetivos

1.1.1 Objetivo General

- Poner en marcha un sistema automatizado para el control de préstamo de equipos odontológicos en el consultorio y mejorar su rentabilidad económica.

1.1.2 Objetivos Específicos

- Controlar el control de flujos de agua, aire y electricidad al equipo para que los usuarios no los usen después de haberse consumido el tiempo total de uso.

- Diseñar un sistema que guarde estadísticas sobre las horas alquiladas a cada máquina para llevar una mejor contabilidad del negocio.

1.2 Justificación

En Cartagena hay muchas empresas que usan la tecnología para mejorar sus margenes de ganancia, sin embargo, en esta ciudad no hay muchas empresas que brinden soluciones tecnológicas para empresas pequeñas que tienen problemas graves, como este consultorio. Este proyecto es una demostración de que se pueden usar los conocimientos adquiridos para desarrollar soluciones prácticas para mejorar las utilidades de

empresas pequeñas en Cartagena.

Capítulo 2.
Metodología y Desarrollo

La metodología a usar sera la metodología para las tenologías de la información y la comunicación de Vaishnavi y Kuechler mostrado en la Figura 2.1.

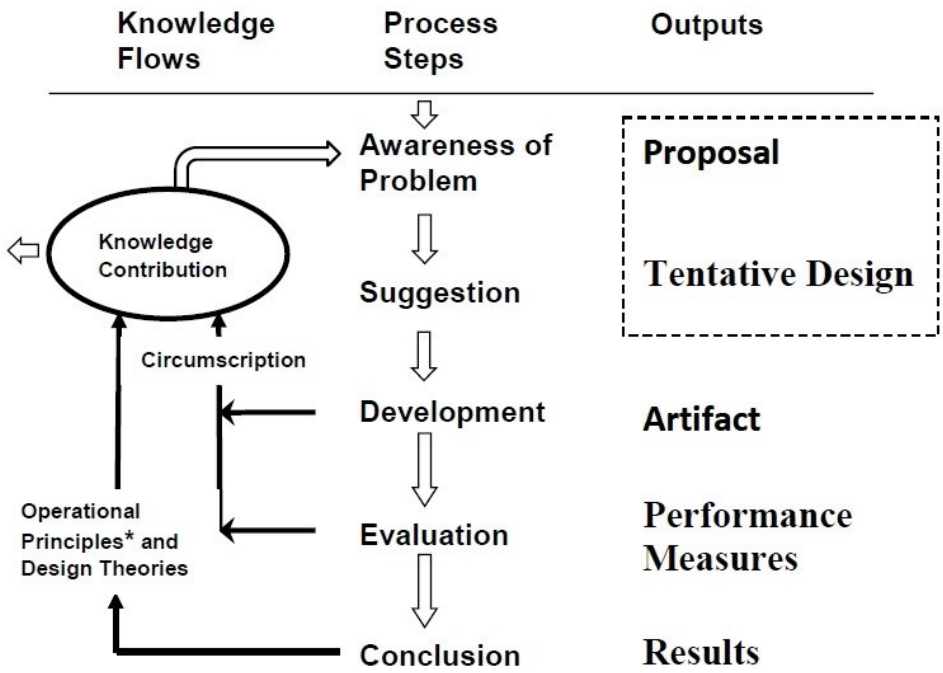


Figura 2.1: Tomado de "Desing Science Research Methods and Paterns: Innovating Information and Communication Technology" [12]

El proceso comienza cuando se descubre un problema y se busca toda la información referente a ese problema, luego se propone una solución mejorando las fallas del sistema. El siguiente paso ya plantea una solución técnica con base a los recursos

que se manejen y al presupuesto con el que se cuenta. Inmediatamente despúes pasa a fase de desarrollo donde se crean prototipos con base a los diseños planteados anteriormente y se evalúa su desempeño. Finalmente se escriben las conclusiones con todos los resultados y principios operacionales desarrollados a lo largo de la investigación.

2.1 Problema

2.1.1 Problema del consultorio

El problema de este consultorio nace por el lugar donde se encuentra, le brinda privacidad al cliente pero carece de supervición por parte de los propietarios. Otra parte del problema son las ambigüedades al momento de alquilar las unidades de trabajo dental porque no se registra una hora exacta de encendido del dispositivo y los usuarios se aprovechan de eso alegando haber encendido el dispositivo despúes tiempo despúes de haber entrado al consultorio. Por otra parte, la única forma para observar la actividad del consultorio estando dentro de la casa de los propietarios es por medio de una ventana y desde esta solamente se observa uno de los equipos, el otro equipo no se puede ver y puede estar siendo usado por algún usuario inescrupuloso que no haya pagado por esta y haya entrado con la excusa de servir de auxiliar a otro usuario que haya pagado por el servicio. Estos son casos mas críticos expuestos por el propietario a los cuales les quiere encontrar solución. En resumen, una lista con los problemas mas sérios es las siguiente:

- Los usuarios no detienen su trabajo inmediatamente se acaba el tiempo, ellos

Figura 2.2: Foto del consultorio visto desde el patio de la propiedad

empiezan a recoger y a veces hasta siguen haciendo el trabajo. Incluso hay casos de usuarios que agendan el consultorio y llegan temprano para aclimatizarlo, pero su verdadera intención es atender a alguien antes de que su tiempo comience.

- Los tiempos de traslado del sitio donde esté el operario hasta el consultorio son tiempos muertos que a lo largo del año suman mucho tiempo perdido.

- Los usuarios pagan por un tiempo en un equipo y usan 2 equipos sin pagar. Esto es por el simple hecho que las sillas solo tienen un interruptor que está a la vista y puede ser manipulado por cualquier persona. Se han visto casos donde el usuario que hace esto paga por varias horas incrementando el desfalco en las finanzas del consultorio.

Todos estos problemas juntos generan un aumento considerable en los gastos del consultorio disminuyendo la utilidad del negocio porque en este establecimiento el tiempo de uso de las máquinas es el unico activo que produce dinero. En la tabla 2.1 se aprecia el dinero producido por cada unidad dental y en la tabla 2.2 se muestran las

pérdidas aproximadas de el sistema. Finalmente en la tabla 2.3 se muestra que se esta perdiendo un poco menos del 20% de las ganancias actuales por estos problemas.

Tabla 2.1: Dinero generado por las sillas

GANANCIAS ACTUALES			
HORAS TRABAJADAS POR DIA	DIAS TRABAJADOS AL MES	VALOR DE LA HORA	GANANCIA POR SILLA
8	20	$ 10.000	$ 1.600.000

Tabla 2.2: Pérdidas mensuales por silla

Promedio de clientes por día	Tiempo perdido por cliente (min)	Costo de silla por hora	Perdidas por cliente anuales
4	15	$ 10.000	$ 300.000

Tabla 2.3: Ganancias del sistema con sus pérdidas actuales.

GANANCIAS ACTUALES			
TOTAL DINERO PRODUCIDO	PERDIDAS ACTUALES	GANANCIAS TOTALES	PORCENTAJE DE GANANCIAS
$ 3.200.000	$ 600.000	$ 2.600.000	81,25%

Además de esto, los encargados de arrendar las unidades son parte de la familia y por tanto dividen su tiempo entre labores domésticas y prestar el servicio de alquiler. Este desorden dificulta llevar registros de horas usadas, registros de dinero, de hecho nisiquiera llevan horas de las personas dentro del consultorio las cuales no detienen su trabajo y esperan a que los corran luego de haber usado mucho mas tiempo que

el pagado. Todo esto genera un estrés mental en el encargado de turno que a veces no compensa con el dinero que el consultorio está generando.

2.1.2 Solución

La solución a estos problemas es un sistema autónomo que permita controlar el tiempo de manera por un operario enviando información de manera remota para que el sistema active o desactive la maquinaria dental por el tiempo pagado, y si el cliente requiere más tiempo de servicio, este deberá acercarse al encargado de momento y pagar por más tiempo. Así se evita que el operario de turno tenga que ir hacia el consultorio a revisar los usuarios no estén excediendo los límites del servicio. El sistema también debe llevar un registro del alquiler de las sillas por día para luego poder analizar las ventas y poder llevar una contabilidad más formal con la que el dueño del sitio pueda analizar sus ingresos, egresos, e inclusive hacer análisis para inversiones futuras.

La función del sistema es controlar el tiempo de uso cortando el fluido eléctrico de los equipos del consultorio cuando el tiempo se acabe. El sistema debe ser automático para que el operario no tenga la necesidad de ir al consultorio todo el tiempo a revisar que todo esté en orden. Tampoco se debe hacer grandes modificaciones que dañen la integridad física del consultorio, porque al ser un establecimiento médico cumple unas condiciones de sanidad exigidas por el ente regulador en la ciudad.

2.2 Planteamiento y diseño del sistema

Las sillas de odontología como las mostradas en la Figura 2.3 funcionan activándoles un switch (*véase Figura* 2.4) en la base de estas. Este switch simplemente cierra y abre el fluido eléctrico hacia la silla activando todos sus sistemas internos. Las sillas internamente tiene unas electroválvulas que manejan cierran los flujos de agua y aire a alta presión, elementos necesarios para el trabajo de los dientes como agua y aire a presión.

Figura 2.3: Silla odonotológica estandar

Figura 2.4: Switch para activar la silla

Actualmente la solución para que los usuarios dejen de trabajar es cortándo el aire a presión, pero no siempre es efectivo ya que a veces dejan de usar el aire a presión y se dedican solo a analizar la boca del paciente. El recurso principal de trabajo es la luz del sistema porque la unidad apagada cierra sus sistemas internos. Por lo tanto, para que los usuarios dejen de trabajar se debe cortar la energía electrica hacia las sillas y así se verán obligados a retirarse o pedir mas tiempo.

El sistema que solucione el problema debe enviar información de manera remota por parte del operario y ser recibida por los equipos en cuestión para cortar la electricidad, entonces a primera vista se sabe que deben tener un dispositivo manipulado por el trabajador del consultorio y otro dispositivo que ejecute las órdenes en el consultorio de manera automática. También debe mostrar el tiempo restante al cliente

para que éste sepa que debe terminar pronto o pagar por mas tiempo.

Por recomendación de un cliente se agregaran 2 electroválvulas que corten el aire presión y el agua porque, aunque la máquina lo hace automáticamente cuando se corta la luz, a veces los estudiantes solo necesitan usar la luz para ciertos trabajos, y se puede pagar un costo menor por la hora del servicio. Se consulto esta aplicabilidad al dueño del consultorio y este aceptó.

En la Figura 2.5 se aprecia lo que hay debajo de estas cajas no es muy complejo, solo son las mangueras que se pueden cortar y colocar las válvulas para cortar los fluidos entrantes a la máquina.

Para la electricidad se puede colocar un TRIAC que funcionará como interruptor eléctrico para la silla. Tambíen se puede utilizar un relé, pero estos son un poco mas costosos y su vida útil también disminuye al gastarse los contactos. La carga, al no ser inductiva, no presenta peligros para usar el TRIAC y se puede aislar el dispositivo de control usando un optoacoplador.

La tecnología RFID es sencilla de usar y entre sus aplicaciones más comunes esta la de cartera electrónica, como la ya mencionada para los párques temáticos o los sistemas de transporte masivos. En este caso particular se tendrá una cartera electrónica local en la cual se guardará tiempo en los tags para activar las sillas. El presupuesto de la tecnología RFID es considerablemente mas bajo que utilizar una infraestructura de red, porque simplemente se carga la tarjeta con información desde el sitio de venta y el usuario lleva esta información hasta el sitio de aplicación. Para

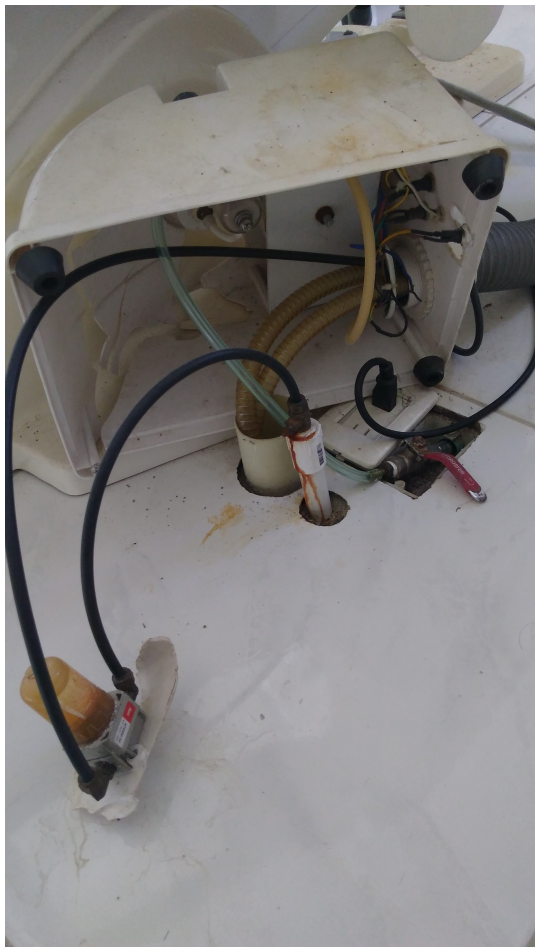


Figura 2.5: Las conexiones debajo del switch.

aplicaciones de RFID el módulo mas común es el MRFC522(*véase Figura* 2.8) que es muy comercial y tiene muchas librerías para muchos microcontroladores y esto facilita su uso para diversas aplicaciones.

La forma mas accesible de usar la tecnologia RFID es usar tarjetas MIFARE[6] las cuales poseen un chip *contactless* (que no se tocan) el cual interactúa con muchos módulos en el mercado, incluyendo el MRFC522 a usar en el proyecto. Estas tarjetas inteligentes se denominan *tags* las cuales básicamente son dispositivos de

almacenamiento de datos, de manera estructurada. Este almacenamiento está dividido en sectores y cada uno de estos sectores esta dividido en bloques, todo esto para garantizar un control de acceso seguro a la información guardada en el tag. La durabilidad y bajo costo de estas tarjetas las ha llevado a ser muy usadas como billeteras electrónicas, control de acceso e incluso tiquetes para sitios de atención masiva como estadios, salas de concierto, entre otras.

Las tarjetas MIFARE mas comunes de encontrar son las de 1Kb y estas ofrecen 1024 bytes de almacenamiento, los cuales están divididos en 16 sectores. Cada uno de estos sectores están divididos en 4 bloques de 16 bytes cada uno en los que se almacena la información necesaria para que el sistema funcione, en la Figura 2.6 se puede apreciar la distribución. Cada sector esta protegido por dos claves A y B las cuales protejen operaciones de lectura, escritura, incremento de valores (Si el bloque tiene formato de valor) y otras para acceder y cambiar información de la tarjeta. Estas claves se almacenan en el cuarto bloque de cada sector y cada sector debe ser autenticado antes de realizar cada accion en el tag, para garantizar la seguridad del sistema.

Como ya se ha mencionado antes, en estas tarjetas la operaciones programadas por defecto son leer, escribir, incrementar valor, entre otras. Para ejecutar cada una de estas funciones en un tag se deben saber la clave del sector a usar y se debe autenticar dependiendo de los bits de acceso programados con anterioridad y guardados en el cuarto bloque de cada sector, llamado tambien *trailerblock*. Estas

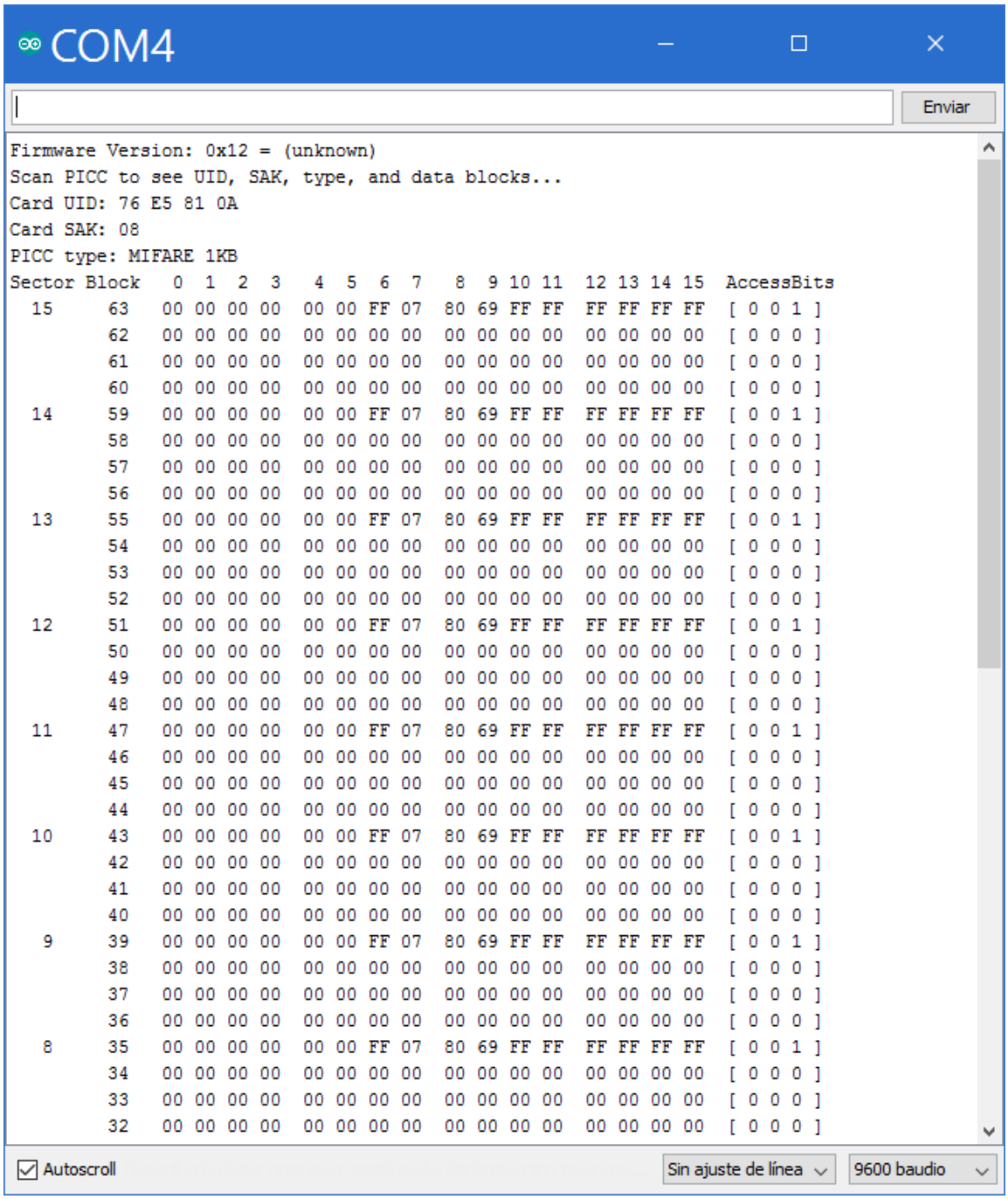


Figura 2.6: Aqui se muestra la distribución de memoria de una tarjeta MIFARE de 1Kb.

tarjetas tienen un sistema de seguridad moderado que no es fácilmente vulnerado por personas con conocimientos bajos en el tema.

Una gran ventaja de estas tarjetas, además de guardar información general, es que permiten manipular formatos numéricos directamente sin necesidad de hacer conversiones de byte a *int* o *long* y modificarlos en ese mismo momento dentro del tag. Ádemas de tener muchas formas creativas de tags para guardar información de forma no molesta y acceder a sistemas de manera rápida y segura.

(a) Tags en forma de *wristband* (b) Tags con forma de llavero (c) Tag con forma de tarjeta

Figura 2.7: Diferentes formas de *tags* MIFARE

Para programar el sistema nos basamos en tarjetas de desarrollo Arduino[1], las cuales tienen un entorno de desarrollo de fácil trabajo y también tienen una librería especializada para usar el módulo RC522. Para que el tamaño del prototipo no sea tan abultado se utilizó el microcontrolador ATMega328 el cual es el microcontrolador que utiliza la tarjeta Arduino UNO y se programa muy facilmente utilizando una de estas placas de desarrollo[3].

Para programar el microcontrolador sin la tarjeta se le debe conectar un cristal

Figura 2.8: Tarjeta MRFC522 [9]

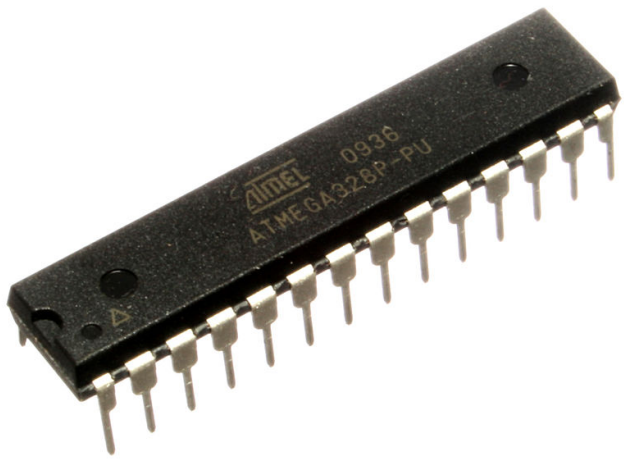
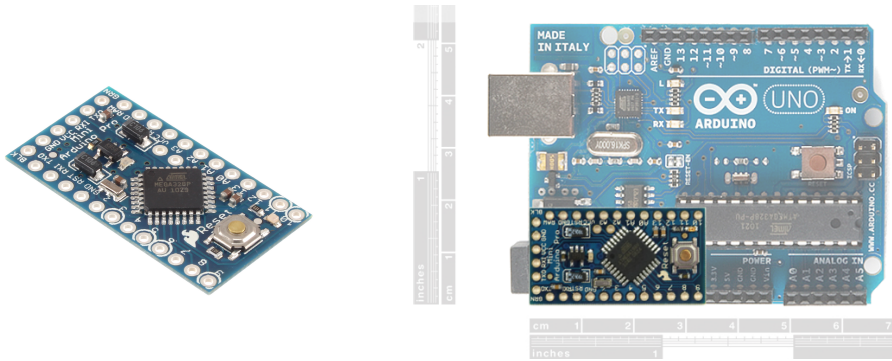


Figura 2.9: ATMega328P - PU

oscilador de 16MHz en los pines 9 y 10. Se puede programar usando un arduino UNO que no tenga el microcontrolador conectado haciendo conexiones entre los pines RST, GND, RX y TX[5] . Para hacerlo mas fácil se tiene una Shield que hace estas



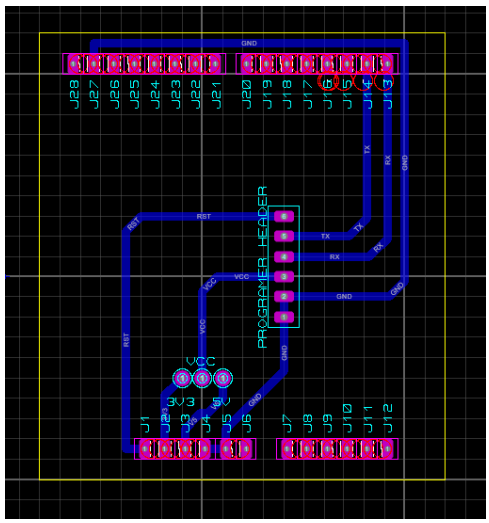
(a) Tarjeta Arduino Mini Pro

(b) Aquí se puede apreciar el tamaño reducido de la tarejta arduino mini pro

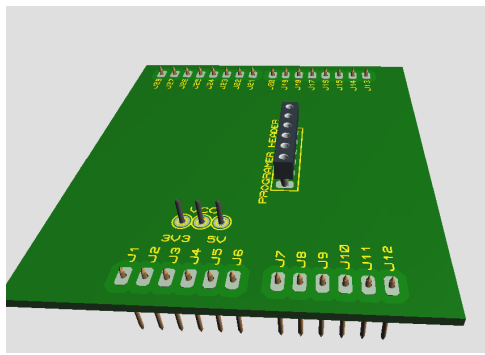
Figura 2.10: Arduino Mini Pro

conexiones y las deja en unos pines con el orden del Arduino mini pro, de esta forma se puede usar el arduino UNO para programar ambos microcontroladores. En las Figuras 2.11 se muestra su diseño desde la PCB y su uso en la vida real.

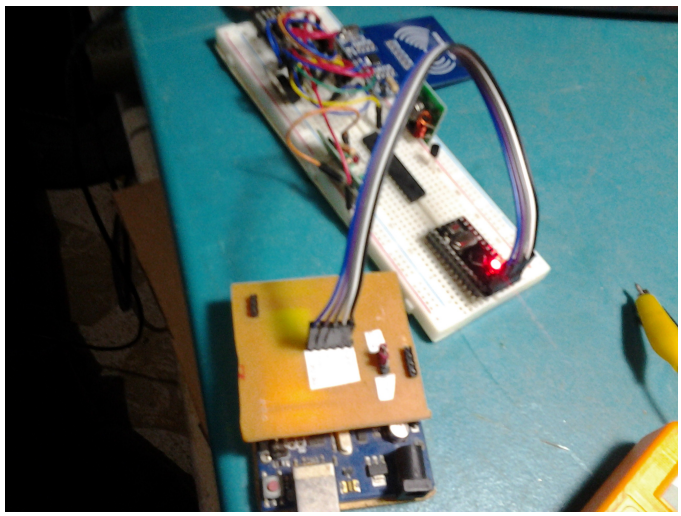
Entonces se van a desarrollar dos dispositivos: uno que lo tenga el operario pueda manipular de manera fácil e intuitiva para enviar la información al otro dispositivo. Este dispositivo será llamado *Porgramador de tarjetas*. El otro dispositivo será el que active los equipos dentales dependiendo de la información enviada por el encargado de turno. Este dispositivo derá llamado *controlador del válvulas*. Para mejorar la seguridad del sistema se utilizarán los dos claves de cada tag y se cambiarán en un bloque específico que tendrá toda la información. Si algún usuario intenta acceder a la informacíon no podrá a menos que tenga las dos claves. El programador de tarjetas debe tener un menú muy sencillo el cual posee las funciones necesarias de seguridad para que el sistema no sea vulnerado con facilidad. Este menú sera mostrado en una pantalla LCD 16x2(*véase Figura* 2.12) y se podrá navegar a través del mismo usando



(a) Diseño de PCB de Shield Programadora



(b) Diseño de en 3D de Shield Programadora



(c) Shield Programadora

Figura 2.11: Shield programadora

un teclado de 2x2 integrado en el dispositivo[7].

Figura 2.12: Pantalla LCD 16x2

El menú será como se muestra a continuación.

- **Grabar tiempo:** Esta sección de menú esta diseñada para programar las horas a alquilar en la tarjeta. Se puede escoger que se va a utilizar de la silla y también dará el costo final del servicio. (*véase Figura* 2.18)

  Esta función toma el sector 2 (el cual debe estar previamente formateado para el sistema) y le escribe valores numéricos en sus bloques 1 y 2 los cuales corresponden a el tiempo programado y los servicios de la silla que activará el sistema. Antes de sumar los valores correspondientes a la venta en proceso se ejecuta una subfunción que borra cualquier valor dentro del tag y los deja en cero, para luego sumarle los valores actuales de la venta, si los valores ya son cero solo suma el valor de la venta. Despúes de haber escrito estos valores en el tag, este dispositivo guarda el valor de la venta y lo suma a un acumulador en la memoria EEPROM de tal forma que vaya llevando un registro de las ventas del día.

- **Formatear:** Esta sección sirve para ingresar tarjetas nuevas al sistemas porque

estas cuentan con unas claves de acceso que generaran seguridad al sistema. También tiene una subfuncioń con la cual se programará una tarjeta master con la que se puede acceder a la silla de manera ilimitada.(*véase Figura* 2.19)

Esta función solo sirve si la tarjeta es nueva, el autentica con la clave por defecto de las tarjetas y la cambia para usar la clave de autenticación del sistema. Ademas de eso le da formato de valor a el bloque 1 y 2 del sector que se este trabajando, para este sistema solo es necesario usar el sector 2.

- **Revisar:** Opción que permite al operario revisar si la tarjeta pertenece al sistema.

  Esta función simplemente autentica y revisa el formato de valor en el sector 2, esta funcion esta incluida dentro de todas las otras, pero se agrega al menu para no hacer todo el proceso de las otras funciones.

- **Vaciar tarjeta:** Opcion que permite al operario quitar todos los valores agregados en una tarjeta.

  Esta función resta los valores actuales de los bloques 1 y 2 de el sector 2 del tag y los deja en 0, de esta manera se evita que se sobreescriban valores en el tag y se alquile tiempo de mas.

- **Revisar ganancias:** Esta seccion de menú muestra las ventas del día y tiene la función de cambio de día con la que se reinicializan estos datos.(*véase Figura* 2.20)

Esta función es un acumulador que va guardando un registro de todas la ventas del día y las muestra. Con esta pequeña ayuda será posible llevar una contabilidad mas organizada si los encargados no borran los datos.

- **Formato master:** Esta funcion permite escribir en un tag del sistema el formato master para ser usada por el dueño del consultorio.

  Esta función escribe en el bloque 0 sector 2 una serie de caracteres distintivos con los cuales la silla se activarśin limite de tiempo, privilegio único para el propietario cuando desee usar la silla para algún procedimiento personal.

- **Serial Dump:** Función que muestra el *dump* de el sector 2 del tag RFID solo si esta conectado al computador con el puerto serial abierto.

- **Borrar master:** Esta función borra el formato master de un tag para ser usada por un usuario común y corriente.

  Esta función borra en el bloque 0 sector 2 en caso tal de que se deba usar el tag master para alquilar una silla de forma normal.

El controlador de válvulas tomará la informacion presente en el tag y luego activará el equipo dental mostrando el tiempo transcurrido con unos displays de 7 segmentos, y cuando el timepo esté a punto de acabarse hará sonar un buzzer para recordar al usuario que debe recargar mas tiempo o prepararse para acabar el trabajo. En caso tal que el usuario necesite mas tiempo este podrá acercarse al operario y pedir una nueva recarga y colocando la nueva tarjeta en el sistema evitará que

este se apague y podrá seguir con su trabajo sin problemas. Adicionalmente habrá una tarjeta con una firma "master" la cual activará el sistema por el tiempo que esta tarjeta este presente, será usada cuando el dueño decida usar el sistema para si mismo sin tener que alterar las estadisticas de dinero grabadas en el dispositivo programador.

2.3 Desarrollo

Esta parte se dividirá en 2 dispositivos y su integración como sistema.

2.3.1 Programador de tarjetas

Diseño

Se desarrolla a base de un microcontrolador ATMega328 el cual se programa con una pantalla LCD de 16x2 servirá como interfaz gráfica para observar las diferentes opciones de menú para los operarios del sistema. También se colocan 4 botones con los cuales se puede por el menú y acceder a las opciones. Su suministro de energía será mediante un adaptador AC/DC a 5v 1A.

Para ser un dispositivo práctico no debe ser muy grande y debe ser sencillo para que cualquiera pueda usarlo dentro del consultorio y debe ser intuitivo para que cualquiera pueda manejarlo sin ningún problema. Con esto se garantiza que cualquiera pueda usarlo, si se llega a dar el caso en que el dueño deba pagarle a otra persona para cuidar el consultorio esta persona no necesariamente debe tener una formación especial, los cual disminuiría el costo de contratación en un futuro. A

pesar de ser ligero y práctico este dispositivo también debe ser robusto para que no sea dañado con facilidad y tenga la durabilidad deseada.

Diseño PCB

Lo primero es diseñar el dispositivo con base a los tamaños del LCD y el RC522 para que no sea tan grande y no se vea desproporcionado, luego se le agregan los botones y el regulador de voltaje para el RC522 y los pines de programación. El diseño final de PCB se puede apreciar en la Figura 2.13.

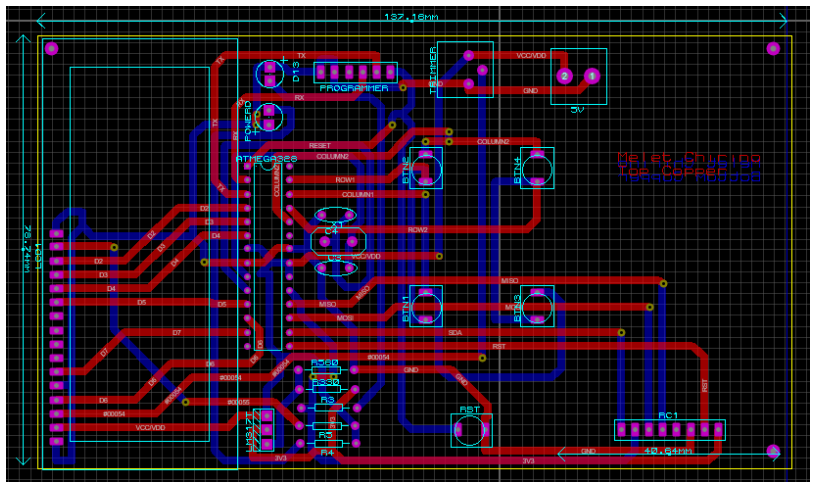


Figura 2.13: PCB del prototipo numero 1

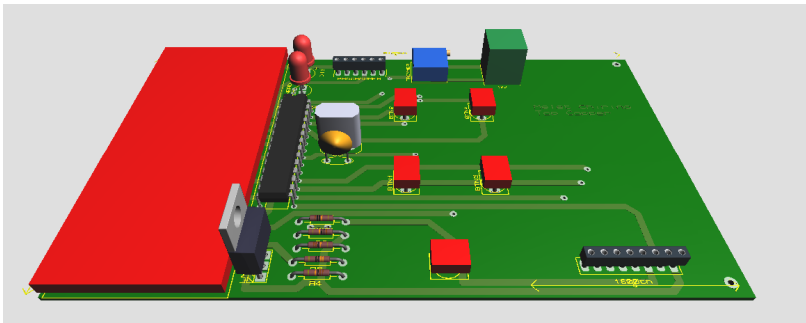


Figura 2.14: Imagen 3D de como debe quedar el dispositivo programador

Este diseño se fabrica en un centro especializado porque al ser una baquelita compleja con una duración deseada de varios años no es recomendable fabricarla por los metodos caseros tradicionales. En la Figura 2.15 se muestra el programador de tarjetas con sus componentes ya soldados y listo para funcionar.

Figura 2.15: Dispositivo programador soldado antes de colocarle la carcaza

Programación

Para programar el microcontrolador utilizando el programa *Arduino IDE* se le debe quemar un *bootloader* antes, sino no será reconocido como una tarjeta de desarrollo. En este bootloader se le asignan diferentes propiedades al microcontrolador que pueden ser usadas despúes.

Para quemar el bootloader se utiliza un Arduino UNO ya funcional y se le graba el programa *ArduinoISP* cargado en los ejemplos por defecto del programa. Luego se conectan los dispositivos como se muestra en la Figura 2.16. Luego en el menú **Herramientas** → **Programador** se selecciona la opción **ArduinoISP**. Finalmente en ese mismo se selecciona la opción **Quemar Bootloader** y esperar a que se queme en el microcontrolador[2].

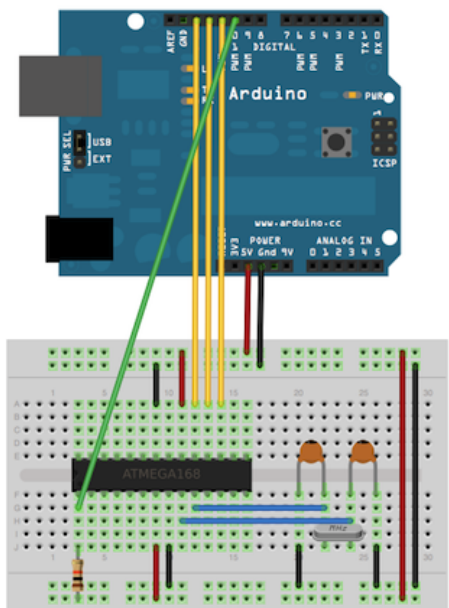


Figura 2.16: Conexiones para quemar bootloader en un arduino

Con el bootloader quemado se puede programar el microcontrolador directamente usando el programa de Arduino. Para programarlo se usa un Arduino UNO sin el microcontrolador, solo debes conectar los pines RX, TX, GND, RESET y 5V en caso de ser necesario. En este punto se utiliza la shield mostrada anteriormente en la Figura 2.11 y se conectan los pines en un cabezal de programación dejado a un lado del circuito.

Finalmente se le programan las funciones que este dispositivo va a ejecutar.

Lo primero es darle un formato a los tags nuevos grabando las claves usadas por el sistema en un sector específico y colocando dos bloques de este sector con formato de valor para que en este se puedan efectuar operaciones numéricas. Luego la función de grabar tiempo simplemente aumenta los valores en los bloques antes mencionados para que sean luego leídos por el segundo dispositivo. Los diagramas de bloques de las funciones de principales de este dispositivo se muestran a continuación.

Los registros de ventas se guardarán en la memoria EEPROM para evitar que se pierdan en caso de desconectarse. Esta memoria tiene 100.000 ciclos de lectura y escritura lo cual disminuye la vida útil del microcontrolador. Es recomendable cambiar el microcontrolador una vez al año para garantizar el funcionamiento adecuado del sistema.

En el loop principal servirá para navegar por las funciones del programador usando el teclado matricial. Se puede ver el funcionamiento de este loop en la Figura 2.17.

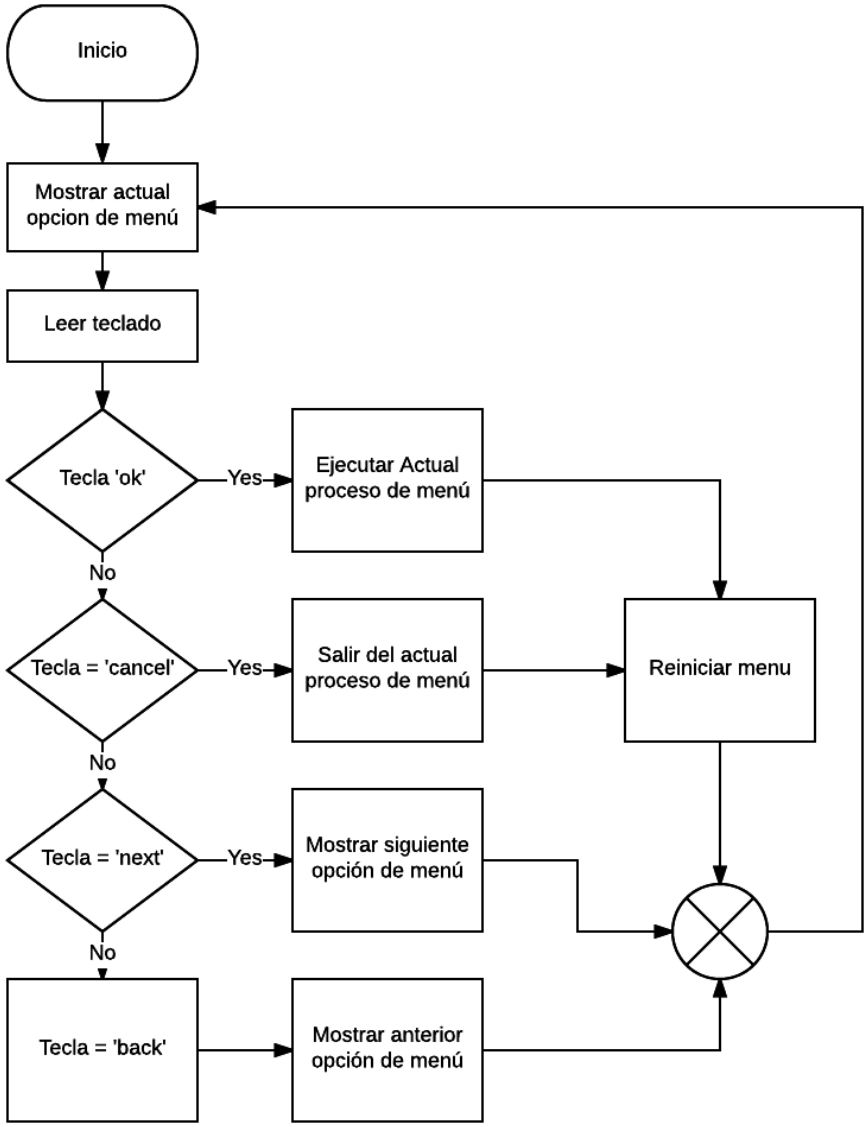


Figura 2.17: Funcionamiento principal

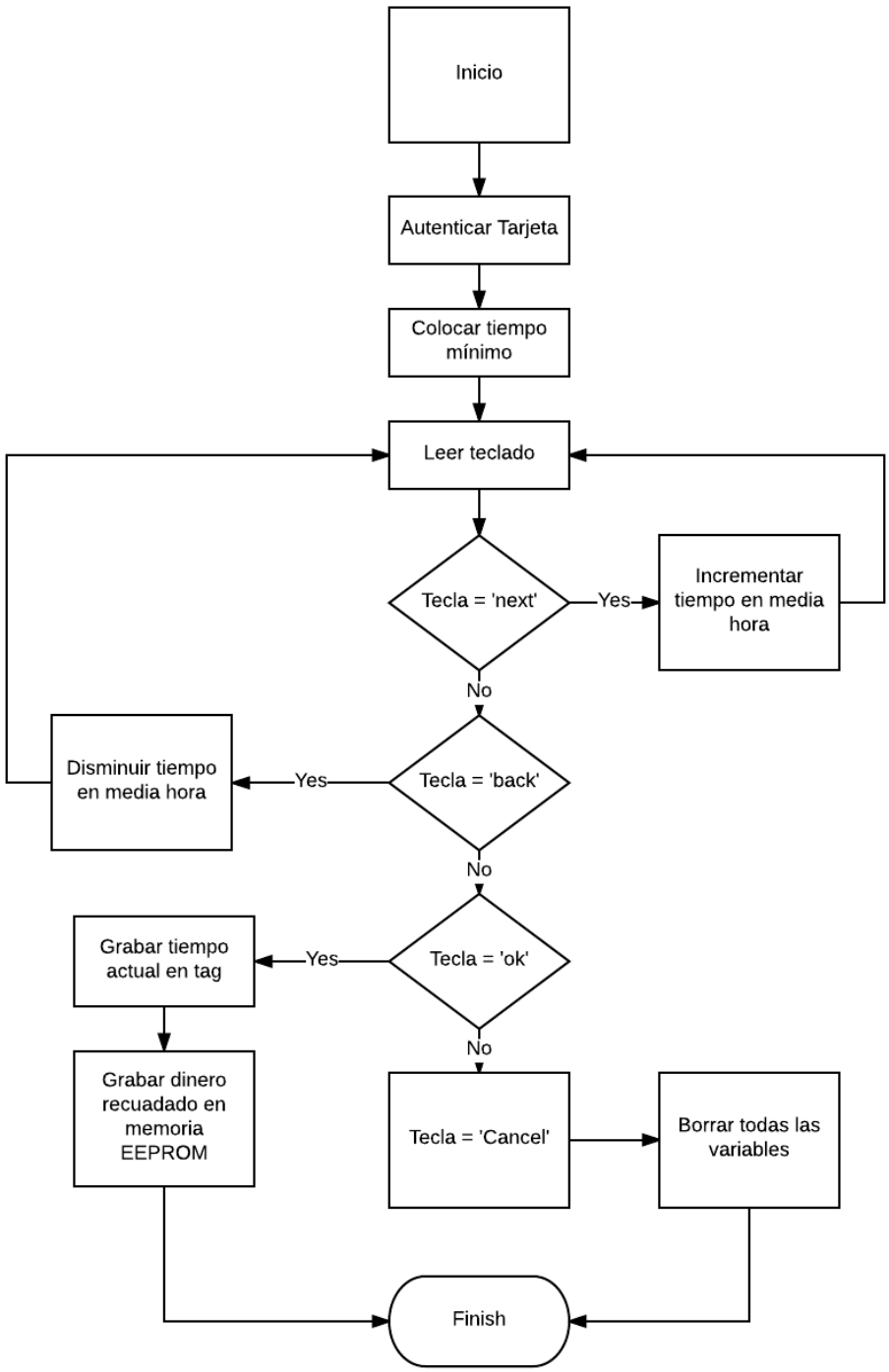


Figura 2.18: Funcionamiento de la opción para recargar

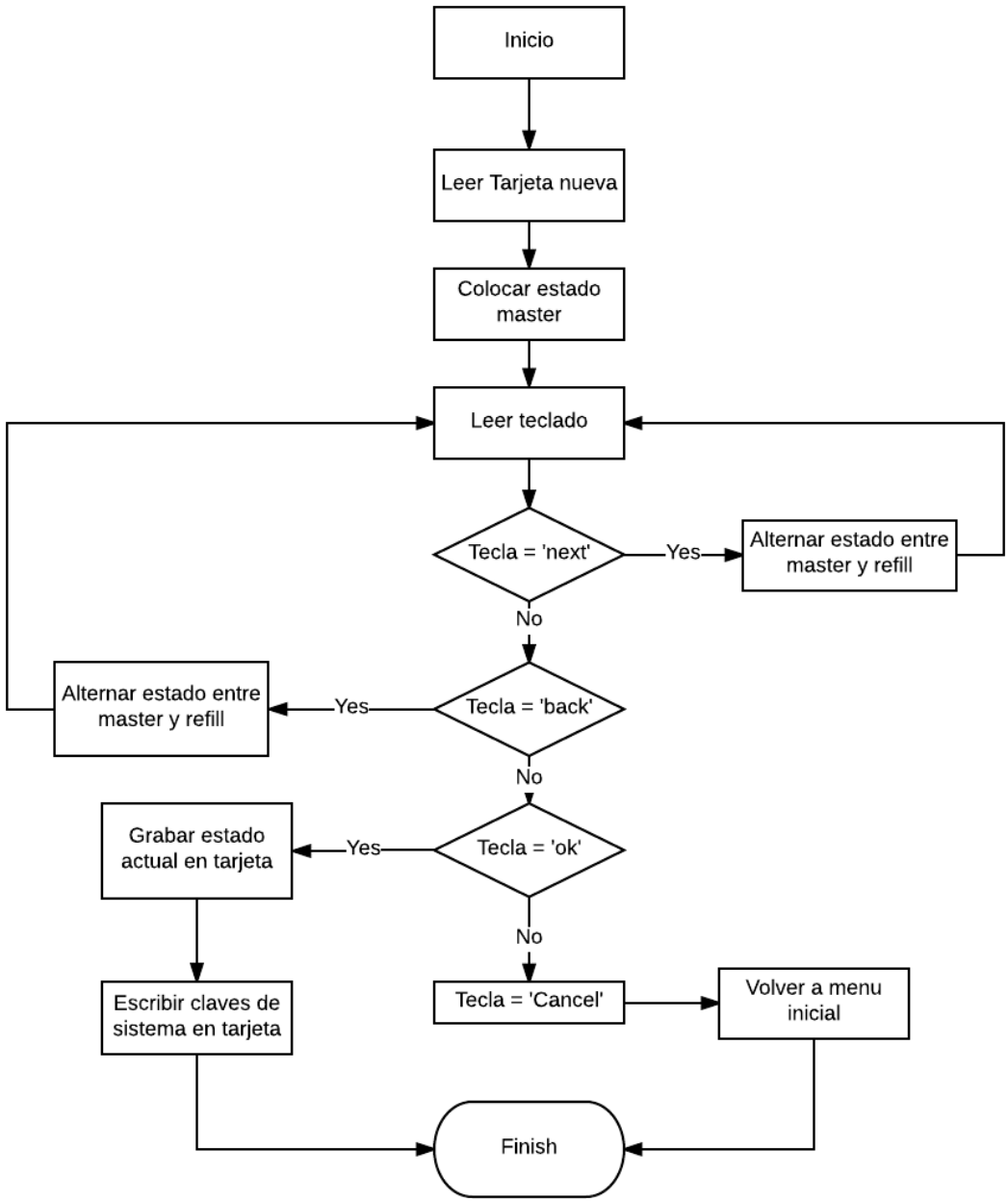


Figura 2.19: Función tarjeta nueva

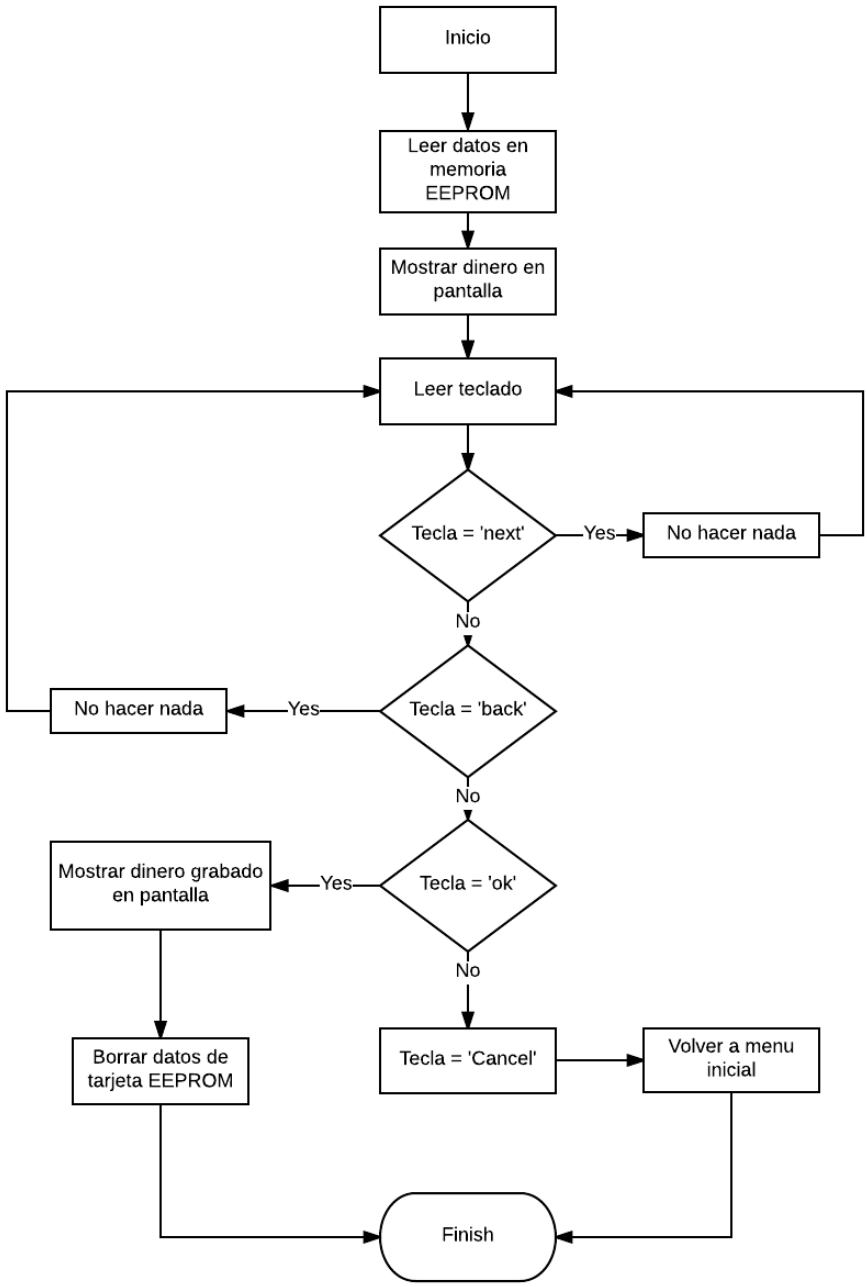


Figura 2.20: Funcion cambio de día

Diseno físico

Acorde al diseño establecido anteriormente, el programador de tarjetas no puede ser muy grande, pero debe ser robusto para aguantar maltrato en caso de desadaptados. Lo mas sencillo para esto es hacer una caja por parte y luego imprimirla en una impresora 3D. La impresión esta dividida en 3 partes: tapa inferior, tapa superior y carcaza media. Estas 3 partes serán atornilladas para quedar juntas. En las Figuras 2.21, 2.22 y 2.23 están cada una de las partes del diseño de las tapas y en la Figura 2.24 se muestra el diseño final.

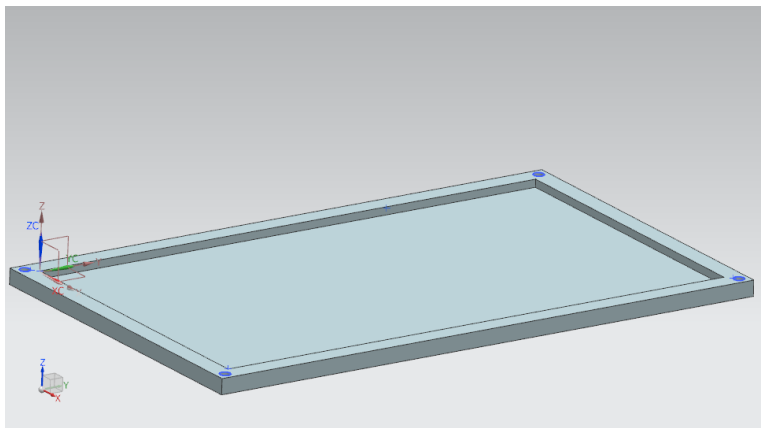


Figura 2.21: Carcaza inferior

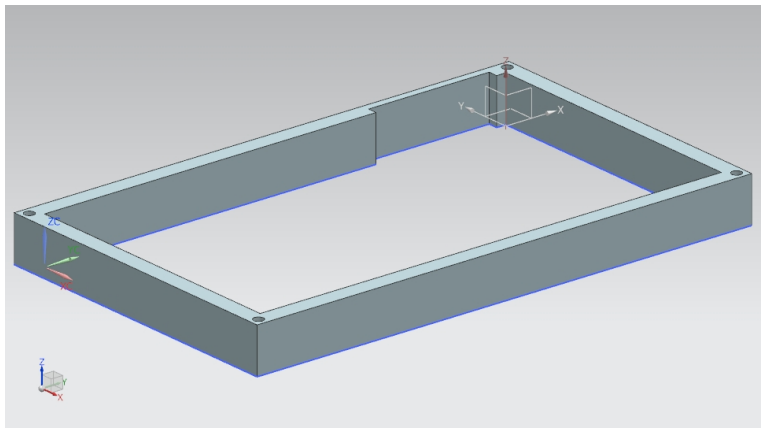


Figura 2.22: Carcaza media

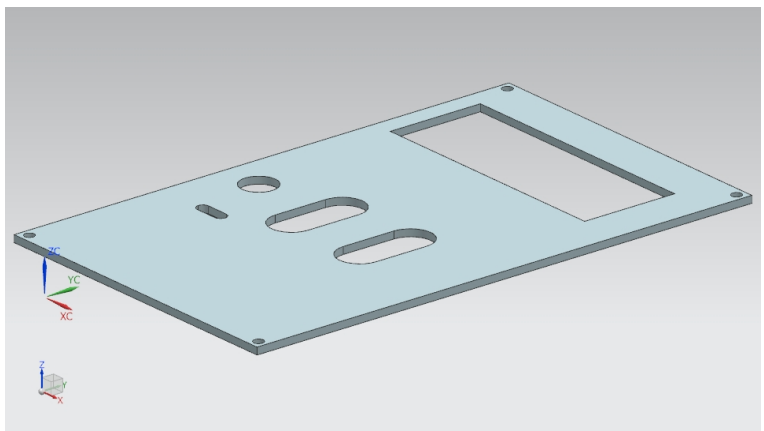


Figura 2.23: Carcaza superior

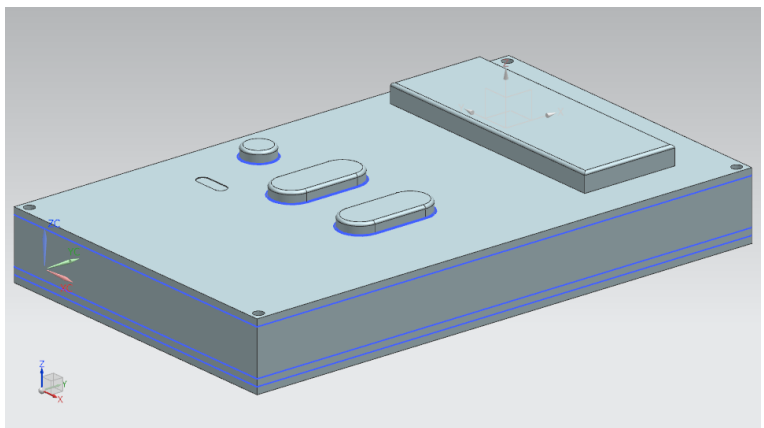


Figura 2.24: Carcaza Finalizada

Luego de haber impreso las piezas se ensamblan y se aseguran con tornillos. El resultado se muestra en la Figura 2.25. El dispositivo ahora esta mas firme y no se desbarata con facilidad. Se le diseñaron unos unos botones superficiales con los que se puede manipular fácilmente el dispositivo. El programador de tarjetas tiene unos LEDs con los que se sabe si está funcionando. Para darle un toque mas estético se forra con papel contact y su vista se mejora considerablemente. El dispositivo final se muestra en la Figura 2.26.

Figura 2.25: Dispositivo programador con su carcaza puesta y listo para ser usado.

Figura 2.26: Dispositivo programador de tarjetas esperando para ser usado en el consultorio.

2.3.2 Controlador de válvulas

Desarrollo

Para este dispositivo es necesario tener ciertas consideraciones como la potencia usada por los dispositivos, los componentes a usar, la manera en como va a interactuar con el usuario, y utilidades adicionales para facilitar su uso.

Lo mas importante en este dispositivo es que pueda encender y apagar la silla dependiendo del tiempo pagado, y al usar un TRIAC es necesario saber la potencia que este puede manejar y por cuanto tiempo. En la Figura 2.27 se puede ver la información eléctrica de las sillas.

El voltaje a usar es de 110VAC y la corriente máxima es $3.5A_{RMS}$, para estas especificaciones se puede usar un TRIAC de la serie BT137-600 los cuales soportan mas de 400VAC y un máximo de 8A de corriente RMS. Estos voltajes de corriente

Figura 2.27: Información electrica de las sillas

alterna son muy peligrosos para el microcontrolador por lo tanto es necesario aislarlo de el TRIAC para evitar dañar el mismo. Al tener un ángulo de disparo de 180°se puede manipular con un optoacoplador MOC3010 y esto nos garantiza un control ON/OFF seguro.

Figura 2.28: Triac BT136-500

El único inconveniente de este TRIAC es la potencia disipada podría recalentarlo causando efectos no deseados en el sistema. Normalmente para evitar esto se atornilla un disipador de calor al circuito integrado, pero necesitamos saber que tanta potencia

debe disipar. Para saber la potencia disipada por el TRIAC observamos la gráfica 1 de la hoja de datos (*véase fig.* 2.29) donde se observa la potencia disipada en función de la corriente RMS. En esa Figura se puede ver que el TRIAC disipa un poco menos de 4W lo cual esta bien porque con esta potencia disipada el TRIAC no supera los 125°C por esa razón no es necesario que el disipador de calor sea tan grande.

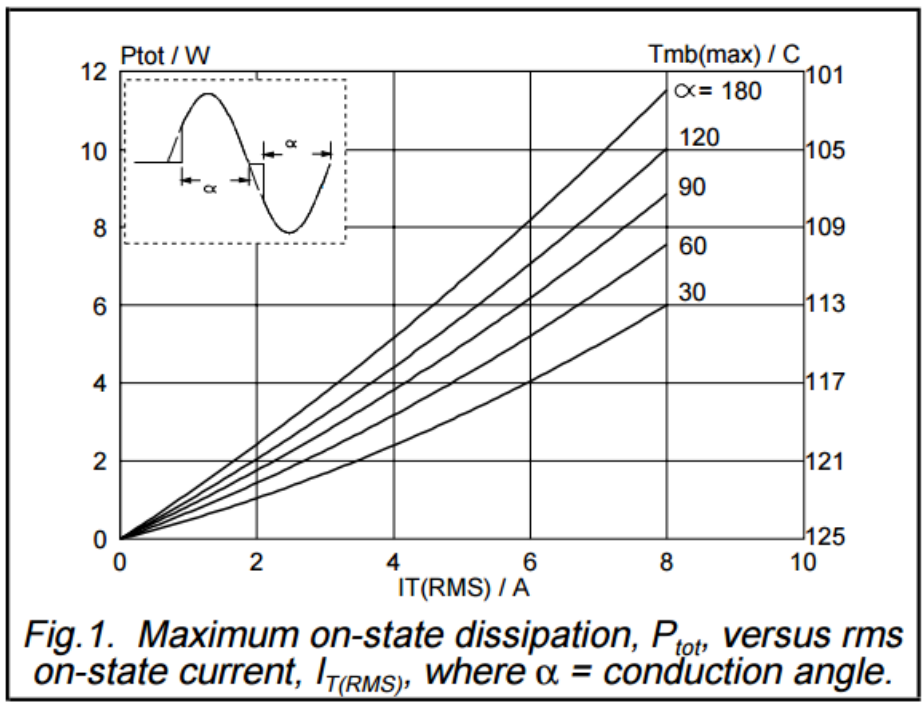


Figura 2.29: Máxima disipación en estado estacionario contra corriente en valores RMS, Tomado de la hoja de datos del TRIAC [10]

Las electroválvulas de 12V serán controladas por transistores TIP31 los cuales soportan hasta 3A de corriente en el colector que es suficiente para encender las electroválvulas por tiempos largos. Las corrientes que pasarán por las electrovaávulas no son tantas como para que sea necesario un disipador de calor en estos transistores.

A diferencia del dispositivo programador de tarjetas, este dispositivo se diseñará por bloques para tener cada parte (potencia, controlador, interacción) cerca del sitio donde necesita ser usada, por ejemplo, la parte de potencia va conectada al toma debajo de la unidad y la parte que lee las tarjetas tiene que estar en alguna parte visible para el trabajador. Otra razón para diseñar esto por módulos es que el sistema a futuro puede ser mejorado y no es necesario que se cambien todos los módulos, solo el que necesite una mejora. Asimismo, otra posible ventaja sobre el trabajo por módulo es un mantenimiento mas fácil, si un dispositivo tiene una avería no es necesario cambiar toda la tarjeta, solo se cambia el módulo que tenga el problema, abaratando el costo de mantenimiento futuro. En la Figura 2.30 se muestran los 3 módulos de este dispositivo.

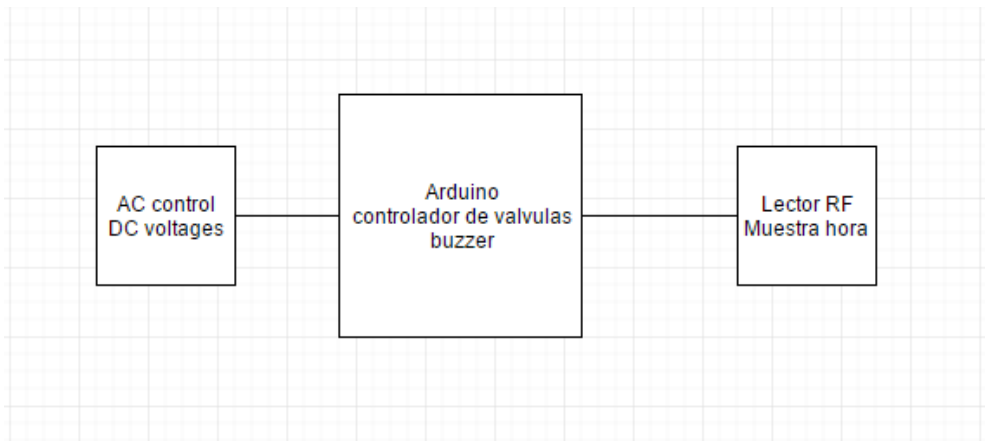


Figura 2.30: Bloques de diseño de dispositivo 2

Este dispositivo funciona leyendo los tags nuevos, solamente toma los datos y deja de usar el RC522 para ahorrar energía mientras se pone en modo de trabajo. Cuando detecta una tarjeta adiciona una hora al reloj de trabajo del dispositivo,

al cabarse esta hora revisa si la tarjeta sigue allí y adiciona otra hora, así hasta que la tarjeta es removida y el sistema deja de adicionar tiempo. Al detectar otras tarjetas aceptadas por el sistema el dispositivo activa el TRIAC y dos electroválvulas, dependiendo de la información en la tarjeta, durante el tiempo establecido, luego apaga las electroválvulas y desconecta el triac para que se apague el sistema. las electroválvulas las enciende con transistores TIP31.

Para mostrar la hora es recomendable utilizar displays de 7 segmentos y su driver para ahorrar pines, y se conectan de tal forma que se pueda usar un multiplexor con la intención de usar menos pines. Estos display de 7 segmentos muestran el tiempo faltante. Cuando falten 5 minutos puede haber una alarma sonora que avise al usuario que el tiempo esta a punto de acabarse.

Programación

Este dispositivo tiene el mismo protocolo de seguridad que el dispositivo programador para que las tarjetas sean compatibles con el sistema, en la Figura 2.31 se ve el funcionamiento del loop principal del controlador de válvulas.

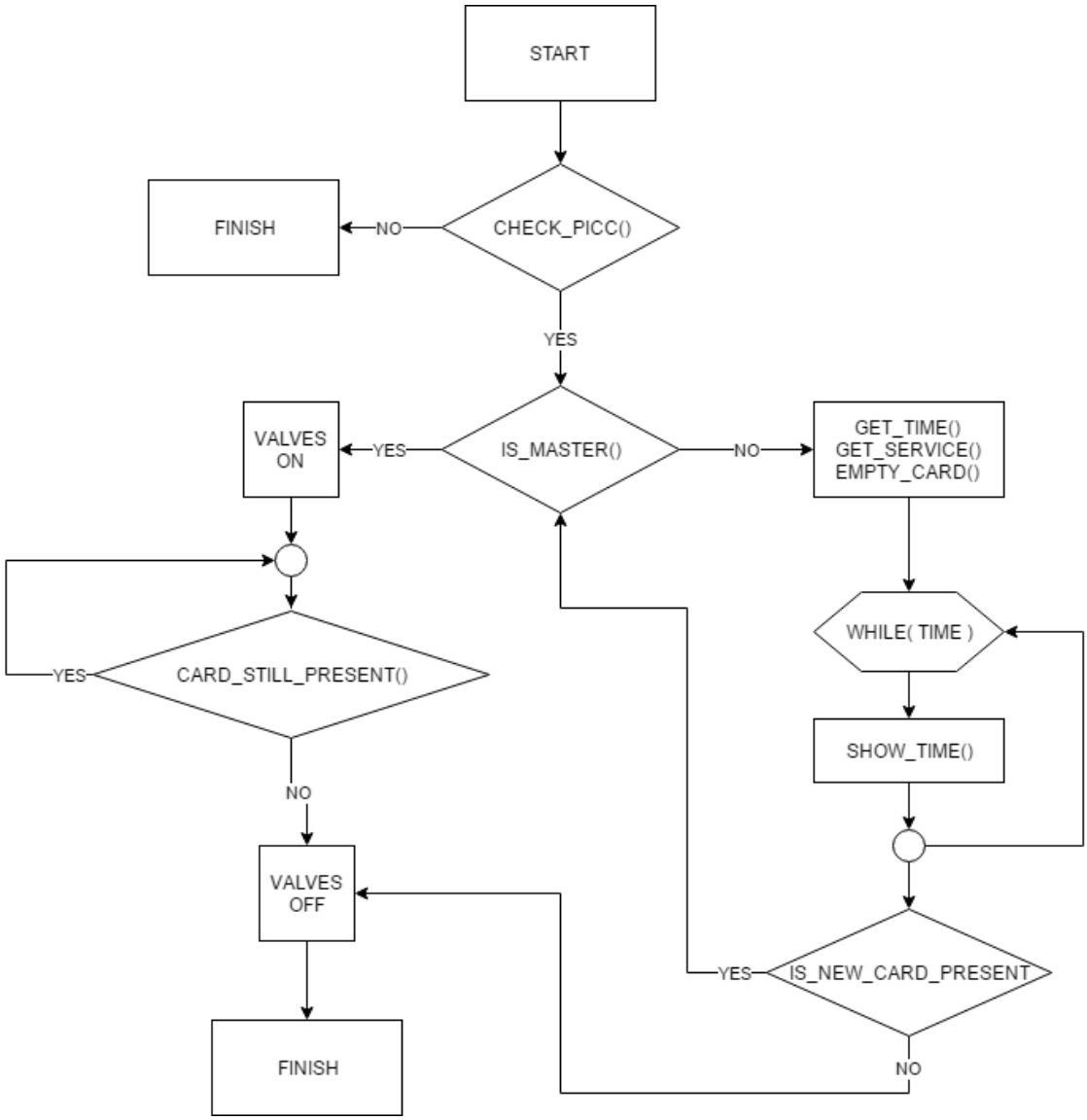
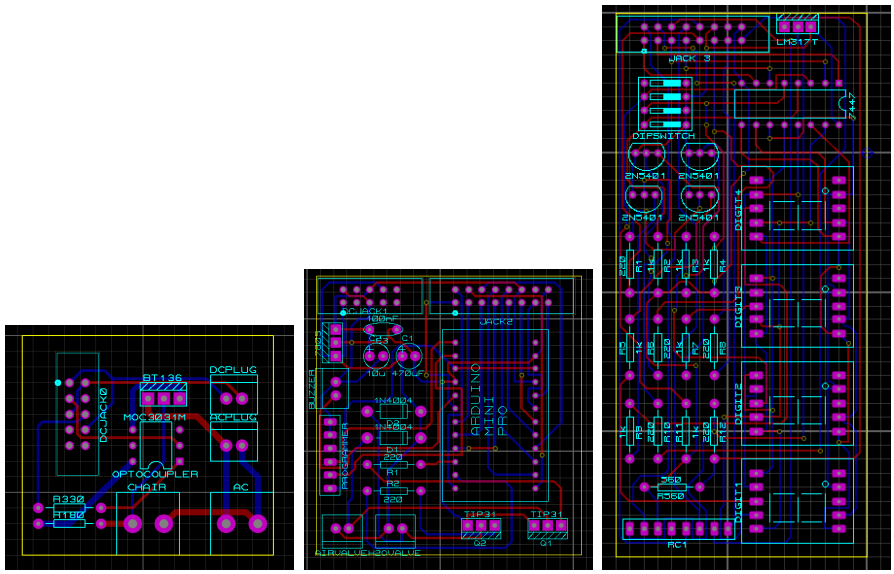


Figura 2.31: Diagrama de flujo de funcionamiento del controlador de válvulas

Básicamente este dispositivo solamente revisa si la tarjeta tiene la firma de master, si es así solo activa el sistema hasta que la tarjeta sea removida; sino es master solo toma el tiempo guardado en la tarjeta y activa el sistema durante este tiempo.

Diseño de PCB

Se hacen los diseños de cada módulo con los que va a trabajar el sistema y se dejan las conexiones para luego conectarlos mediante un bus de datos. En la Figura 2.32 se aprecia el diseño de PCB.



(a) Bloque de AC (b) Bloque de Arduino (c) Bloque de interacción

Figura 2.32: Diseños de PCB de cada uno de los módulos.

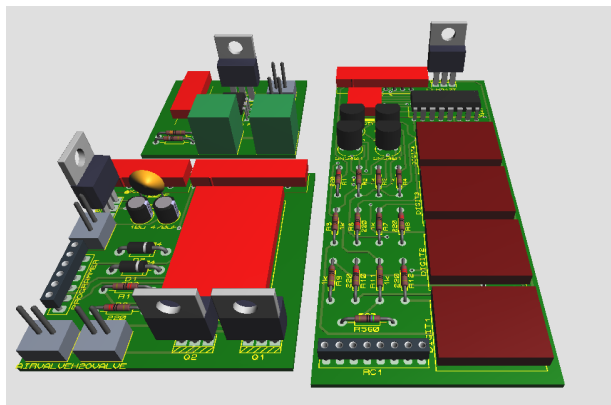


Figura 2.33: Imagen en 3D de los módulos del dispositivo controlador de vávulas.

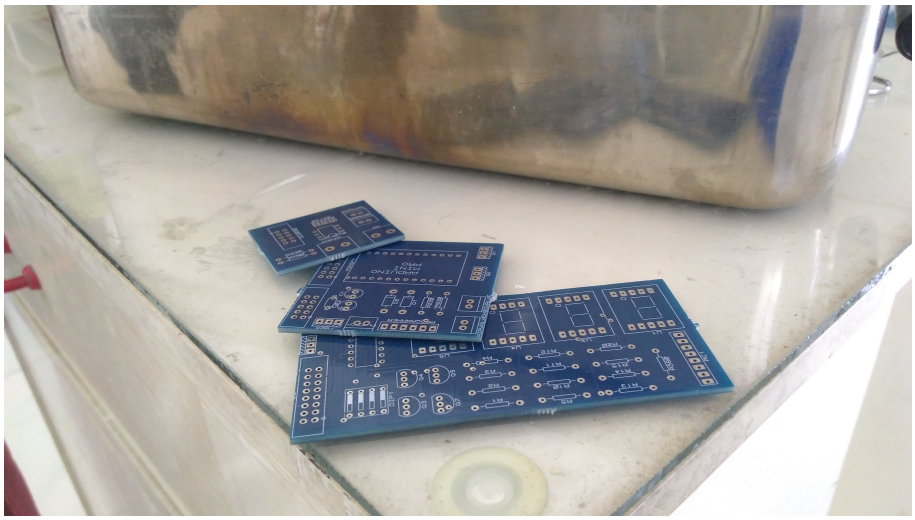


Figura 2.34: PCB fabricadas

Diseño y fabricación de carcaza

La naturaleza modular del dispositivo nos obliga a diseñar diferentes carcazas, pero la mas importante es la carcaza del módulo de interacción porque este es el que va a estar a la vista de todos, los otros dos van a ir debajo de la unidad(*véase fig.* 2.5) y estarán tapados por una tapa, no es necesario hacer una carcaza cerrada pero se le puede poner una base de sujeción para que no estén en el suelo. También es recomendable dejar estos dispositivos ventilados y que mejor ventilación que dejarlos en un entorno abierto. En la Figura 2.35 se ve la base de los dos módulos. En las imágenes 2.36 2.37 y 2.38 se muestra el diseño de la carcaza del módulo que va afuera.

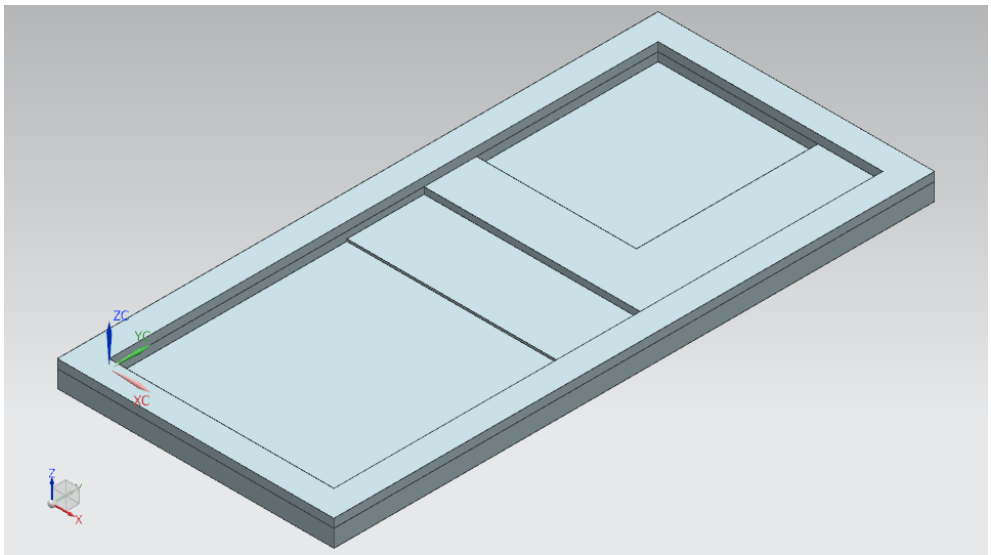


Figura 2.35: Base sujetadora de los módulos ocultos bajo la silla.

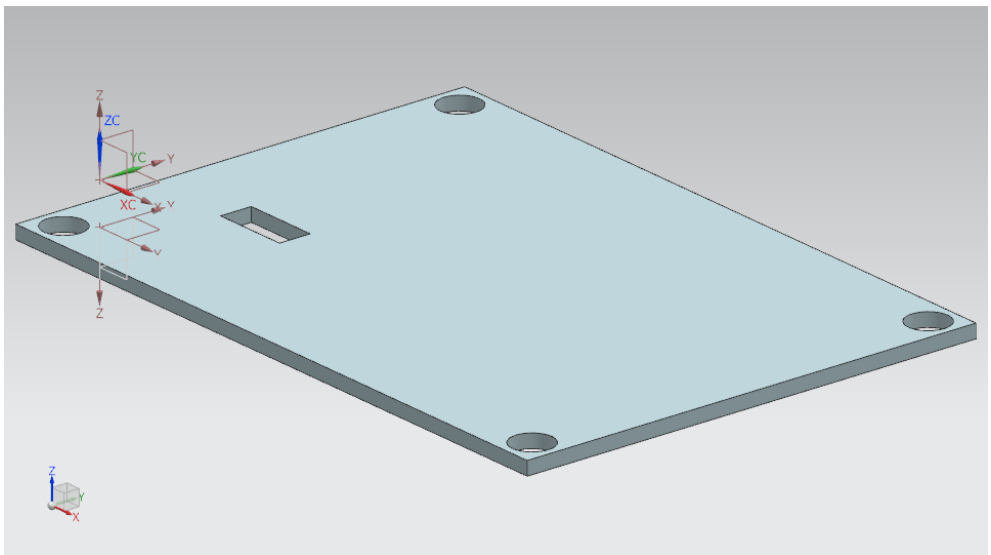


Figura 2.36: Carcaza inferior del bloque de interacción

Finalmente en la Figura 2.39 se ve el dispositivo final conectado en sus últimas pruebas de desempeño.

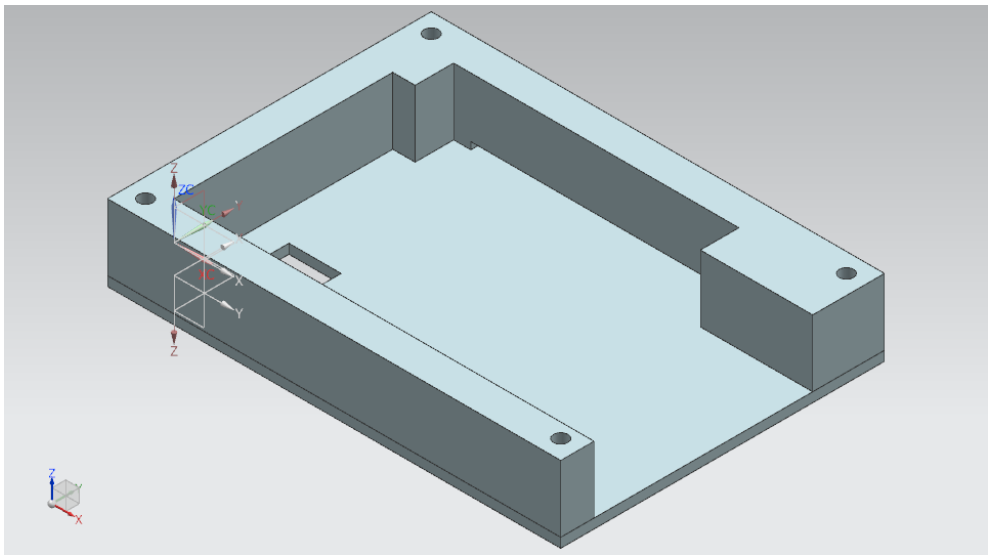


Figura 2.37: Carcaza media del bloque de interacción

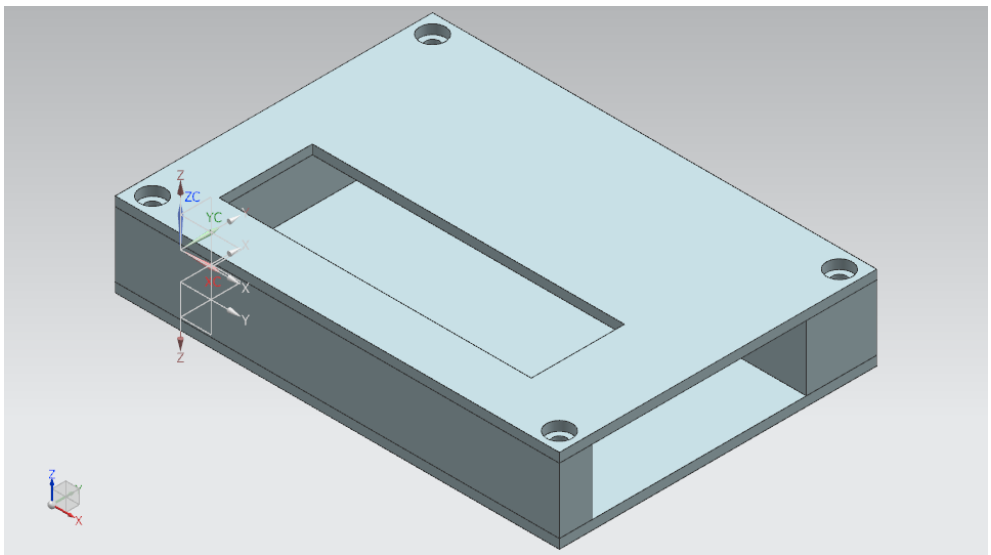


Figura 2.38: Carcaza completa del bloque de interacción

Programación

La función principal de este dispositivo será leer la información en las tarjetas y activar los sistemas si la información está bien. Si la información no es correcta el dispositivo no va a encender y se mantendrá apagado hasta que se coloque una tarjeta

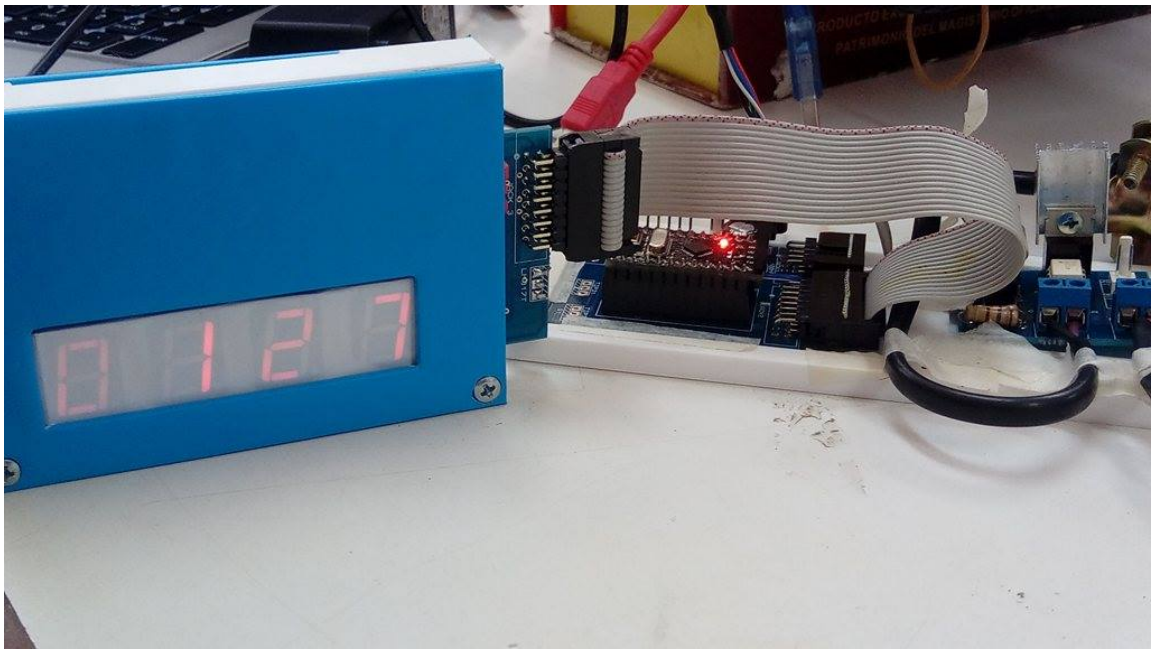


Figura 2.39: Dispositivo controlador de válvulas en sus ultimas pruebas

correcta.

Instalación

Gracias al diseño del dispositivo solo hay que conectar las alimentaciones a la red eléctrica del sitio y el dispositivo enciende de manera correcta. Este dispositivo trabaja solo cuando las tarjetas correctas son colocadas en el sistema, sino no funcionarán. En las imagenes 2.40 y 2.41 se muestran cada uno de los módulos instalados y funcionando.

Figura 2.40: Bloque de dispositivo que muestra el tiempo restante.

2.4 Sistema

En la Figura 2.43 se observa el diagrama de bloques del funcionamiento del sistema. Primero el encargado del sitio graba el tiempo en el tag, luego este tag es llevado al consultorio donde se coloca en el dispositivo mostrado en la Figura 2.40 y este procederá a verificar que la tarjeta pertenezca al sistema y tenga tiempo guardado y avisará por medio de un buzzer si no enciende. Si la tarjeta pertenece al sistema, los displays de 7 segmentos procederán a encenderse de manera automática.

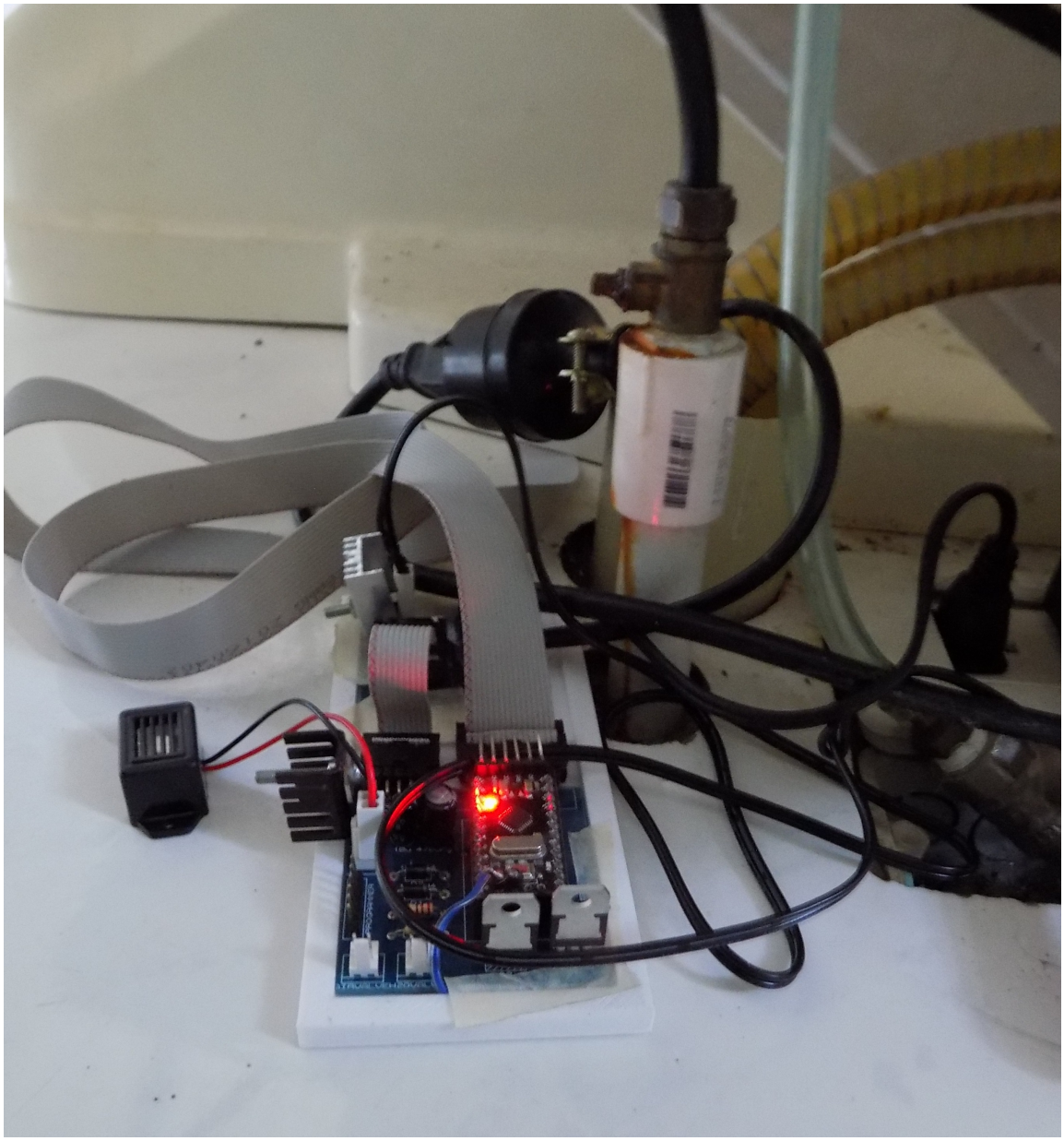


Figura 2.41: Dispositivo oculto bajo la tapa de la unidad de trabajo dental.

Figura 2.42: Unidad de trabajo dental con el dispositivo instalado.

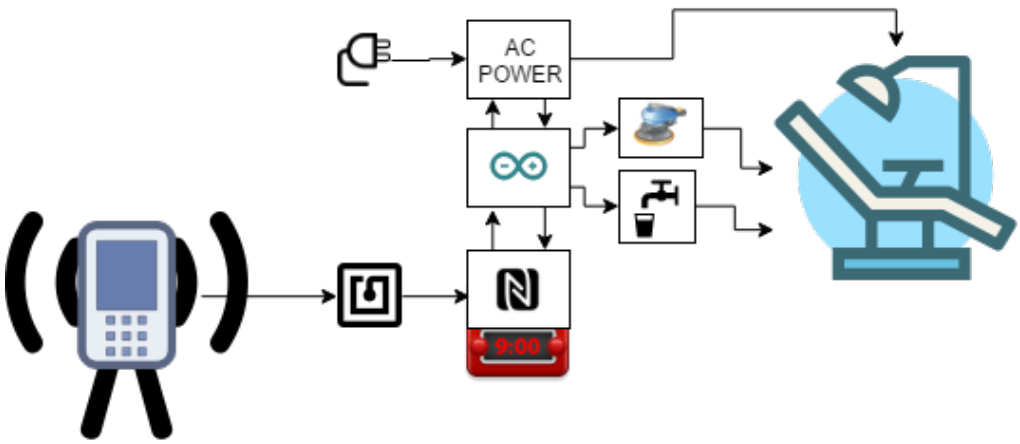


Figura 2.43: Diagrama de bloques del funcionamiento del sistema

Capítulo 3.
Resultados y Discusión

3.1 Rendimiento del sistema

El sistema funciona grabando información en el tag, luego el tag se lleva a la unidad dental y finalmente esta enciende dependiendo de la información grabada en la silla. En el siguiente link se encuentra un video del sistema funcionando https://youtu.be/tcqQ3Mt3cvI.

Una de las consideraciones mas cruciales a tener en cuenta para este proyecto fue que el voltaje en el módulo MFRC522 fuera muy cercano a los 3.3V porque una variación pequeña genera un mal funcionamiento de en este módulo, cortando el canal de comunicación entre los dispositivos y anulando la funcionalidad del sistema. Otra causa de variaciones de voltaje son los 7 segmentos porque al encenderse aumentan la corriente en este módulo y generan un aumento de voltaje en el módulo RFID. Para evitar que esto intervenga de manera negativa en el sistema, los displays de 7 segmentos no se encenderán mientras el módulo MRFC522 esté en funcionamiento. Al momento de autenticar el dispositivo apaga los 7 segmentos, pero cuando termina la comunicación con el tag, el microcontrolador deja de usar el módulo MRFC522

para poder usar los displays de 7 segmentos.

Otra de las consideraciones es que el dispositivo debe tener un pequeño retardo en la lectura de las tarjetas porque hay veces que detecta la tarjeta en una distancia no adecuada y borra la información escrita en esta sin terminar de autenticar, lo que crea un pequeño fallo. Los retardos mas adecuados son de 2 o 3 segundos para que el usuario no tenga necesidad de esperar mucho tiempo a que el sistema autentique y tome la información en la distancia correcta.

Como retroalimentación del dispositivo se utiliza el buzzer, con base a sus sonidos el dispositivo notifica si el tag no autentica, si el tag no tiene hora y el mometno de encendido y apagado de la unidad dental. Es el medio más confiable porque con los displays 7 segmentos el voltaje de el módulo MRFC522 se ve afectado, como ya se mencionó antes.

3.2 Costos de proyecto

3.2.1 Costos de desarrollo

Los costos de desarrollo incluyen el costo de los componentes, el personal que trabajó en el proyecto, alquiler de equipos que se usaron en pro de este proyecto. En la tabla 3.1 se muestra el presupuesto para las personas que trabajaron en el proyecto.

Tabla 3.1: Gastos de personal en el proyecto

NOMBRE	**FORMACIÓN ACADÉMICA**	**DEDICACIÓN (Hrs)**	**EFECTIVO**	**TOTAL**
PhD Juan Carlos Martinez	PhD	10	$ 100.000	**$ 1.000.000**
Melet David Chirino Caicedo	Pregrado Ingenería Mecatrónica	50	$ 25.000	**$ 1.250.000**
TOTAL			**125.000**	**$ 2.250.000**

En las tablas 3.2, 3.3 y 3.4 se muestra el costo de los componentes de los dispositivos.

Tabla 3.3: Materiales de controlador de válvulas

DESCRIPCIÓN	**CANTIDAD**	**EFECTIVO**	**TOTAL**
Capacitor 470uF	1	$ 500	**$ 500**
Capacitor 10u	1	$ 500	**$ 500**
Capcitor 100nF	1	$ 100	**$ 100**
Resistor 220	6	$ 100	**$ 600**
Resistor 1k	7	$ 100	**$ 700**
Resistor 220	1	$ 200	**$ 200**
Resistor 180	1	$ 200	**$ 200**
Resistor 330	1	$ 200	**$ 200**
Resistor 560	1	$ 200	**$ 200**
Transistor TIP31	2	$ 600	**$ 1.200**
Diodos 1N4004	2	$ 100	**$ 200**
Decodificador 7447	1	$ 1.300	**$ 1.300**
Regulador de voltaje 7805	1	$ 1.300	**$ 1.300**
Bornera 2 posiciones	2	$ 800	**$ 1.600**
Pines Molex 2 Posiciones	3	$ 800	**$ 2.400**

Tabla 3.3: Materiales de controlador de válvulas

DESCRIPCIÓN	CANTIDAD	EFECTIVO	TOTAL
TRIAC BT-137 600	1	$ 900	**$ 900**
Bornera 2 posiciones	2	$ 1.000	**$ 2.000**
Regleta de pines doble	2	$ 2.300	**$ 4.600**
Display 7 segmentos	4	$ 2.500	**$ 10.000**
Dipswitch 4 posiciones	1	$ 700	**$ 700**
Regleta de pines doble	2	$ 1.000	**$ 2.000**
Regulador de voltaje LM317T	1	$ 1.600	**$ 1.600**
Arduino Mini Pro	1	$ 17.500	**$ 17.500**
Optoacoplador MOC3031M	1	$ 2.500	**$ 2.500**
Regleta de pines sencilla	1	$ 200	**$ 200**
Sensor RFID RC522	1	$ 30.000	**$ 30.000**
Transistores 2N5401	4	$ 300	**$ 1.200**
Cables Ribbon (mts)	4	$ 5.000	**$ 20.000**
Conectores Ribbon 10 Posiciones	2	$ 700	**$ 1.400**
Conectores Ribbon 16 Posiciones	2	$ 700	**$ 1.400**
Disipadores de Calor	2	$ 1.500	**$ 3.000**
Cables calibre 14 (mts)	1	$ 2.600	**$ 2.600**
Conectores para AC	2	$ 2.700	**$ 5.400**
Fuente de voltaje 12V	1	$ 9.000	**$ 9.000**
TOTAL		**$127.200**	**$ 127.200**

En las tablas 3.5, 3.6 y 3.7 se muestra el costo de otros equipos necesarios para desarrollar el proyecto.

Finalmente en la tabla 3.8 se muestra el costo total del proyecto en el cual se totalizan cada uno de los items para desarrollar el proyecto.

Tabla 3.2: Materiales Programador de tarjetas

DESCRIPCION	CANTIDAD	EFECTIVO	TOTAL
Capacitores 22p	2	$ 500	**$ 1.000**
Resistores 1k	2	$ 300	**$ 600**
Resistor 330	1	$ 300	**$ 300**
Resistores 560	2	$ 200	**$ 400**
LED	1	$ 300	**$ 300**
Bornera 2 posiciones	1	$ 800	**$ 800**
Microcontrolador ATMega328P	1	$ 12.000	**$ 12.000**
Micropulsadores	5	$ 600	**$ 3.000**
Pantalla LCD 16x2	1	$ 23.300	**$ 23.300**
Regulador LM317T	1	$ 1.600	**$ 1.600**
LED	1	$ 1.000	**$ 1.000**
Regleta de pines sencilla	1	$ 1.000	**$ 1.000**
Sensor RFID RC522	1	$ 30.000	**$ 30.000**
Reisstor Variable Preajustable	1	$ 800	**$ 800**
Cristal Oscilador	1	$ 1.000	**$ 1.000**
Fuente de voltaje 5V	1	$ 9.000	**$ 9.000**
Tags RFID	5	$ 2.500	**$ 12.500**
TOTAL		**$ 77.100**	**$ 77.100**

Tabla 3.4: Costo total de los componentes del sistema

DESCRIPCION	CANTIDAD	EFECTIVO	TOTAL
Controlador de valvulas	2	$ 127.200	**$ 254.400**
Programador de tarjetas	1	$ 77.100	**$ 77.100**
TOTAL			**$ 254.400**

Tabla 3.5: Software usado en el proyecto

DESCRIPCIÓN	**COSTO**	**TOTAL**
Proteus 8 Porfesional	$ 150.000	$ 150.000
NX 10.0	$ 300.000	$ 300.000
Dremel Printing	$ -	$ -
Arduino IDE	$ -	$ -
TOTAL	**450.000**	**450.000**

Tabla 3.6: Servicios tenológicos del proyecto

DESCRIPCION	**CANTIDAD**	**MOTIVO**	**EFECTIVO**	**TOTAL**
Colcircuitos	1	Fabricacion de baquelitas	$93.514	**$ 93.514**
TOTAL				**$ 93.514**

El costo final del proyecto tomando todo en cuenta es de $3.631.340. Este es el precio que debe pagar el dueño del consultorio como inversión inicial para adquirir el sistema, además de pagar por un mantenimiento anual de $30.000 para el funcionamiento óptimo del sistema.

3.2.2 Costos de manutención

Adicional a los costos antes mencionados, el sistema tiene un consumo de corriente el cual será un pago realizado mes a mes por el consultorio. El dispositivo programador consume una corriente entre 40mA y 48mA cuando activa la antena asi que tiene una potencia máxima de 240mW. El dispositivo controlador de válvulas tiene una corriente RMS de 13mA por lo que su potencia es aproximadamente 1.5W. En la tabla 3.9 se muestran el costo de manutención por hora y al año suponiendo que se

Tabla 3.7: Compra y arrendamientos de equipos utilizados en el proyecto

EQUIPOS				
NOMBRE EQUIPO	**JUSTIFICACION**	**EFECTIVO**	**CANTIDAD**	**TOTAL**
COMPRAS				
PLA (mts)	Material para imprimir en 3D	$ 652	100	**$ 65.211**
SUB TOTAL COMPRAS				***$ 65.211***
ARRENDAMIENTOS				
Impresora 3D (horas)	Imprimir carcaza	$19.722	10	**$ 197.222**
SUBTOTAL ARRENDAMIENTOS				***$ 197.222***
EQUIPOS DE USO PROPIO				
Multimetro	Instrumento de medicion electronica	$76.500	1	**$ 76.500**
Tarjeta Arduino UNO	Tarjeta usada para programar los microcontroladores	$45.000	1	**$ 45.000**
Cautin	Equipo usado para soldar los componentes sobre las placas electronicas	$23.000	1	**$ 23.000**
SUBTOTAL EQUIPOS PROPIOS				***$ 144.500***
			TOTAL	**$ 406.933**

Tabla 3.8: Costos totales del proyecto

RUBROS		**TOTAL**
GASTOS DE PERSONAL		
DIRECTOR	*$ 1.000.000*	*$ 2.250.000*
ESTUDIANTE	*$ 1.250.000*	
EQUIPOS		
COMPRA	*$ 254.400*	*$ 398.900*
USO	*$ 144.500*	
RECURSOS EXTRAS		
SOFTWARE	*$ 450.000*	*$ 653.514*
MATERIALES E INSUMOS	*$ 110.000*	
SERVICIOS TECNOLOGICOS	*$ 93.514*	
MATERIAL BIBLIOGRAFICO		
SUSCRIPCIONES	*$ -*	*$ 328.926*
LIBROS	*$ 328.926*	
TOTAL		***$ 3.631.340***

trabajen 8 horas al día todos los días. El precio del kiloVatio-hora en esta zona de la ciudad es aproximadamente $427.

Tabla 3.9: Costos de manutención del sistema

Costo de manutencion				
Potencia Controlador de valvulas (W)	Programador de tarjetas (W)	Potencia total Sistema (W)	Costo por hora (kW*h)	Costo Anual
1,4575	1,4	4,315	$ 2	$ 5.378

A este valor inicial se le puede adicionar un costo anual de reprogramación y mantenimiento general del sistema el cual tiene un costo de $30.000 donde se le hacen mantenimiento a las válvulas, se reprograma el microcontrolador en caso tal de subir los precios y se verifican que los cables y conexiones no estén maltratados por el uso. Se aprecia que el costo del sistema por año es considerablemente bajo si tomamos en cuenta la inversión inicial.

3.3 Beneficios del sistema

Este negocio está muy maltrecho por el poco control dentro del consultorio a la hora de alquilar la unidad, la causa mas común de pérdida son las ambigüedades a la hora de adquirir el servicio, no se acuerda una hora precisa de encendido de la máquina porque los usuarios alegan no haber encendido la unidad desde que pagan por el servicio, sino un tiempo despúes. Por este motivo se están perdiendo aproximadamente 10 minutos por cliente, tiempo que a largo plazo deja de generar rubros considerables, como se muestra en la tabla 2.2.

Se observa que una silla pierde aproximadamente el 20% del dinero total que genera. La tabla general de ganancia con las dos sillas funcionando se muestra en la tabla 2.3.

Con el sistema en marcha el costo mensual de las pérdidas se anularía totalmente. Sin embargo, habría que pagar un precio mensual por el sistema por 3 años, que es su vida útil. Este costo mensual se observa en la tabla 3.10.

Tabla 3.10: Costo mensual del sistema a 3 años.

COSTO MENSUAL DEL SISTEMA			
INVERSION INICIAL	AÑOS DE VIDA	COSTO VALOR ANUAL	COSTO SISTEMA POR MES
$ 3.631.340	3	$ 35.378	$ 103.819

Tabla 3.11: Ganancias con el sistema funcionando.

GANANCIAS CON SISTEMA			
TOTAL DINERO PRODUCIDO	PERDIDAS ACTUALES	GANANCIAS TOTALES	PORCENTAJE DE GANANCIAS
$ 3.200.000	$ 103.819	$ 3.096.181	96,76%

En la tabla 3.11 se observa un claro aumento de 15.5% en las utilidades lo cual sobrepasa el objetivo inicial del sistema. Además de incrementar las utilidades el sistema funciona de manera autónoma lo cual significa menos esfuerzo para las personas que trabajan en el sitio, dejando mas tiempo para dedicar a labores domésticas o alguna otra obligación que tengan en el momento.

El sistema también lleva una contabilidad diaria de las ganancias que generen las sillas con lo que se puede llevar una contabilidad mas precisa del consultorio. Con una contabilidad precisa se pueden observar, controlar y analizar gastos, su evolución y distribución respecto a preriodos de tiempo con la intención de mejorar la eficiencia de la empresa y elevar los ingresos en tanto sea posible[4].

Capítulo 4.

Conclusiones y Trabajo futuro

4.1 Soluciones tecnológicas

Este proyecto demuestra que la carrera de ingeniería mecatrónica tiene potencial para desarrollar soluciones tecnológicas para empresas medianas y pequeñas donde la tecnología le asigne tareas repetitivas y tediosas a sistemas autónomos y dejando para los humanos el trabajo de análisis y adicionar valor agregado a sus productos, mejorando la productividad de la empresa[11].

Se puede apreciar claramente en este caso que la tarea de monitoreo de hora quedó y la tarea de llevar el registro de ganancias fue dejada en manos de un sistema, disminuyendo el trabajo del operario, que muchas veces debío dejar de lado sus obligaciones por estar pendiente de las personas que se encontraban dentro del consultorio. Ahora el sistema trabaja solo y las personas que laboran allí pueden dedicarse a pensar en como mejorar su eficiencia, revisar su contabilidad para mejorar ingresos e incluso pensar en futuras inversiones.

4.2 Potencial RFID

El potencial de la tecnologia RFID es inmenso sobretodo en un sistema integrado en el cual se puede manejar una identificación, datos volatiles cada empleado dentro del tag y contadores para controlar tiempos o dineros pagos dentro de empresas.

4.3 Trabajo futuro

Como trabajo futuro se planea montar un sistema completo de control de tiempo dentro del consultorio. Uno de los artefactos que mas consume actualmente dentro del sitio es el aire acondicionado y la gente lo empieza a usar antes durante y despúes de el tiempo pagado. Este proyecto puede ser ampliado para controlar el estado del aire acondicionado para mejorar su uso y disminuir pérdidas dentro de este sitio.

Otro tema a mejorar es el tema de contabilidad, actualmente el sistema instalado permite llevar un registro de la contabilidad, que antes era nula, pero mas adelante este sistema puede llevar una contabilidad por si solo sabiendo exactamente hora de ingresos, dinero recaudado, gastos dentro del consultorio (agua, luz, consumo de aire comprimido, entre otros) y ofreciendo posibles oportunidades de inversión para generar aún mas ingresos.

Capítulo 5.
Anexos

5.1 Códigos

5.1.1 Programador de tarjetas

Los siguientes son los códigos utilizados en los programas. Para correrlos todos deben estar en la misma carpeta y esta se debe llamar "ProgramadorDeTarjetas". Ademas estos codigos necesitan la libreria "MFRC522.h"[8], la libreria "Keypad.h", las otras librerias estan descargadas en el IDE. El siguiente es el loop principal y los otros contienen funciones con las que se ayuda este loop principal.

ProgramadorDeTarjetas.ino

```
//Inicializar el teclado
#include <Keypad.h>

const byte ROWS = 22; // 2 filas
const byte COLS = 2; //2 columnas
//Definir el keymap del teclado
char hexaKeys[ROWS][COLS] = {
  {'1', '2'},
  {'3', '4'}
};
byte rowPins[ROWS] = {18, 16}; //Pines a los que estan conectadas las
    filas del teclado
byte colPins[COLS] = {17, 15}; //Pines a los que estan conectadas las
    columans del teclado
```

```
// Inicializar el objeto teclado
Keypad keys = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

//Inicializar RC522

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN      9          // Configurable, see typical pin layout
    above
#define SS_PIN       10         // Configurable, see typical pin layout
    above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

//Definir objetos key para usarlos para autenticar los sectores de los
    tags RF
MFRC522::MIFARE_Key key;   // Clave por defecto en los tags
MFRC522::MIFARE_Key newKey; //Clave usada en el sistema

//Libreria LCD:
#include <LiquidCrystal.h>

// Definir pines
LiquidCrystal lcd(6, 7, 5, 4, 3, 2);

//Libreria para usar memoria eeprom
#include <EEPROM.h>

// Definir variables de menu
unsigned int sizeMenu = 3;
const byte timeBlock = 9; // bloque donde se guarda el tiempo para que la
    maquina funcione
const byte serviceBlock = 10; // bloque donde se guarda la eleccion de
    servicios

String textoMenu[] = {"Grabar tiempo", "Formatear", "Revisar", "Vaciar
    tarjeta","Revisar Ganancias", "Formato master","Serial Dump","Borrar
    master"};
int earnings = 0;


void setup() {
  inicializarRF522(); //Rutina para inicializar RC522
```

```
  inicializarLCD();   //Rutina para inicializar LCD
}
int menuCursor = 0;

void loop() {
  printLcd(textoMenu[menuCursor], 0, 0, 1);

  int keyLoop = waitForKeypad();

  MENU_SURFING(keyLoop);
}
```

FUNCIONES_TECLADO.ino

```
int char2int(char key) {
  /*
     Esta funcion me pasa los valores de caracter que devuelve el teclado
         a valores
     numericos mas faciles de procesar para el computador
  */
  if (key == '1') {
    return 1;
  }
  if (key == '2') {
    return 2;
  }
  if (key == '3') {
    return 3;
  }
  if (key == '4') {
    return 4;
  }
}

int waitForKeypad() {
  /*
     Esta funcion es una rutina de captura de teclado en la cual me ahorro
         tener que leer
     caracteres y pasarlos a numeros, la funcion lo hace sola.
  */
  char key = keys.waitForKey();
  return char2int(key);
}
```

OPCIONES_MENU.ino

```
void formatPICC() {
  waitForPICC();//Rutina que espera una tarjeta en el lector
  writeNewKey();//Rutina que escribe la nueva clave en la tarjeta


  autenticar('a');//autentica con las claves elegidas en los argumentos
  autenticar('b');//autentica con las claves elegidas en los argumentos
  formatValueBlock(timeBlock);
  formatValueBlock(serviceBlock);
  // Show the whole sector as it currently is
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &newKey, 2);
  Serial.println();
  checkPicc(true);
}


bool checkPicc(bool halt) {
  waitForPICC();//Rutina que espera una tarjeta en el lector
  if (autenticar('a') && autenticar('b')) //autentica con las claves
      elegidas en los argumentos
  {
    printLcd(F("Tarjeta Correcta"), 0, 0, 1);
  } else {
    return false;
  }
  if (checkFormatValue(timeBlock) && checkFormatValue(serviceBlock))
      //autentica con las claves elegidas en los argumentos
  {
    printLcd(F("Valores correctos"), 0, 1, 0);
  } else {
    return false;
  }
  if (!halt)
    return true;
  // Halt PICC
  mfrc522.PICC_HaltA();
  // Stop encryption on PCD
  mfrc522.PCD_StopCrypto1();
  return true;

}
```

```
void grabarTiempo() {
  int serviceSelector = funcionDeServicio();
  int timeSelector = funcionDeTiempo();
  long money;
  if (serviceSelector == 20)
    money = timeSelector * 0.1667;
  else
    money = timeSelector * 0.0833;
  money = money * 1000;
  printLcd(F("Valor a cobrar"), 0, 0, 1);
  printLcd(F("$"), 0, 1, 0); lcd.print(money);
  int  keys = waitForKeypad();
  if (keys == 2)
    return;
  checkPicc(false);
  emptyBlock();
  addValue(serviceSelector, serviceBlock);
  addValue(timeSelector, timeBlock);

  //verificamos revisando el tiempo y los servicios usados
  int finalTime = getValue(timeBlock);
  printLcd(F("Tiempo = "), 0, 0, 1);
  lcd.print(timeFormat(finalTime));

  //printLcd(timeFormat(finalTime), 0, 1, 0);
  // Halt PICC
  mfrc522.PICC_HaltA();
  // Stop encryption on PCD
  mfrc522.PCD_StopCrypto1();
  //Serial.println(money);
  money = money / 1000;
  byte finalMoney = EEPROM.read(earnings);
  finalMoney += money;
  Serial.print("finalMoney = "); Serial.println(finalMoney);
  EEPROM.write(earnings, finalMoney);
  delay(3000);
}

int funcionDeTiempo() {
  int finalTime = 10;
  int keys;
  do {
    printLcd(F("Tiempo"), 0, 0, 1);
```

```
    printLcd(timeFormat(finalTime), 0, 1, 0);
    keys = waitForKeypad();
    switch (keys) {
      case 1:
        finalTime += 10;
        printLcd(F("Solo Luz"), 0, 0, 1);
        break;
      case 2:
        return 0;
        break;
      case 3:
        if (finalTime - 10 <= -1)
          break;
        finalTime -= 10;
        break;
      case 4:
        break;
    }
  } while (!(keys == 4 || keys == 2));
  return finalTime;
}


int funcionDeServicio() {
  int finalService = 10;
  int keys;
  printLcd(F("Solo Luz"), 0, 0, 1);
  do {
    keys = waitForKeypad();
    switch (keys) {
      case 1:
        finalService = 10;
        printLcd(F("Solo Luz"), 0, 0, 1);
        break;
      case 2:
        return 0;
        break;
      case 3:
        finalService = 20;
        printLcd(F("Luz, Agua y"), 0, 0, 1);
        printLcd(F("Aire comprimido"), 0, 1, 0);
        break;
      case 4:
        break;
```

```
    }
  } while (!(keys == 4 || keys == 2));
  return finalService;
}
```

funcionesLCD.ino

```
/*
  LiquidCrystal Library - Hello World

  Demonstrates the use a 16x2 LCD display. The LiquidCrystal
  library works with all LCD displays that are compatible with the
  Hitachi HD44780 driver. There are many of them out there, and you
  can usually tell them by the 16-pin interface.

  This sketch prints "Hello World!" to the LCD
  and shows the time.

  The circuit:
   LCD RS pin to digital pin 6
   LCD Enable pin to digital pin 7
   LCD D4 pin to digital pin 5
   LCD D5 pin to digital pin 4
   LCD D6 pin to digital pin 3
   LCD D7 pin to digital pin 2
   LCD R/W pin to ground
   LCD VSS pin to ground
   LCD VCC pin to 5V
   10K resistor:
   ends to +5V and ground
   wiper to LCD VO pin (pin 3)

  Library originally added 18 Apr 2008
  by David A. Mellis
  library modified 5 Jul 2009
  by Limor Fried (http://www.ladyada.net)
  example added 9 Jul 2009
  by Tom Igoe
  modified 22 Nov 2010
  by Tom Igoe

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/

void inicializarLCD() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
```

```
  // Print a message to the LCD.
  lcd.print("Iniciando.");
  Serial.println(F("Lcd iniciado"));

}
void printLcd(String texto, int num1, int num2, bool erase) {
  /*
     Esta funcion es una rutina para imprimir datos en el lcd en la cual
        ingresas datos
     de cursor, texto y puedes elegir si borras la pantalla totalmente.
  */
  if (erase)
    lcd.clear();
  lcd.setCursor(num1, num2);
  lcd.print(texto);
}
String timeFormat(int timeUsed) {
  if (timeUsed < 60)
    return "00:" + String(timeUsed);
   else {
    int hours = timeUsed / 60;
    int minutes = timeUsed % 60;
    if (minutes == 0)
      return String(hours) + ":" + String(minutes) + "0";

    return String(hours) + ":" + String(minutes);
  }
}
```

funcionesRF.ino

```
/**
   --------------------------------------------------------------------------
   This is a MFRC522 library example; see
      https://github.com/miguelbalboa/rfid
   for further details and other examples.

   NOTE: The library file MFRC522.h has a lot of useful info. Please read
      it.

   Released into the public domain.
   --------------------------------------------------------------------------
   This sample shows how to read and write data blocks on a MIFARE Classic
      PICC
   (= card/tag).

   BEWARE: Data will be written to the PICC, in sector #1 (blocks #4 to
      #7).


   Typical pin layout used:
   -----------------------------------------------------------------------------------
               MFRC522     Arduino      Arduino   Arduino   Arduino
                  Arduino
               Reader/PCD  Uno          Mega      Nano v3   Leonardo/Micro
                  Pro Micro
   Signal      Pin         Pin          Pin       Pin       Pin             Pin
   -----------------------------------------------------------------------------------
   RST/Reset   RST         9            5         D9        RESET/ICSP-5    RST
   SPI SS      SDA(SS)     10           53        D10       10              10
   SPI MOSI    MOSI        11 / ICSP-4  51        D11       ICSP-4          16
   SPI MISO    MISO        12 / ICSP-1  50        D12       ICSP-1          14
   SPI SCK     SCK         13 / ICSP-3  52        D13       ICSP-3          15

*/



byte dataMaster[]  = {
  'j', 'u', 'l', 'i', // 1, 2,  3,  4,
  'a', 'n', 'a', 'y', // 5, 6,  7,  8,
  's', 'i', 'l', 'v', // 9, 10, 255, 12,
  'a', 'n', 'a', 0xff // 13, 14, 15, 16
```

```
};


void inicializarRF522() {

  Serial.begin(9600); // Initialize serial communications with the PC
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  byte help1[] = {'M', 'a', 'r', 'i', 'a', 'n', 'a'}; // Clave de sistema
  byte help[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}; // Clave de
      sistema
  // Prepare the key (used both as key A and as key B)
  // using FFFFFFFFFFFFh which is the default at chip delivery from the
      factory
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xff; //Definir los bytes de la clave por defecto
    newKey.keyByte[i] = help1[i]; // Definir los bytes de la clave usada
        por el sistema
  }

  Serial.println(F("Scan a MIFARE Classic PICC to demonstrate read and
      write."));
  Serial.print(F("Using key (for A and B):"));
  dump_byte_array(key.keyByte, MFRC522::MF_KEY_SIZE);
  dump_byte_array(newKey.keyByte, MFRC522::MF_KEY_SIZE);
  Serial.println();

  Serial.println(F("BEWARE: Data will be written to the PICC, in sector
      #2"));
}
/**
   Helper routine to dump a byte array as hex values to Serial.
*/
void dump_byte_array(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}


void waitForPICC() {
  /*
     Esta funcion espera por una nueva tarjeta para empezar a trabajar
```

```
  */
  while (! mfrc522.PICC_IsNewCardPresent()) {
    printLcd(F("Coloque nueva tarjeta"), 0, 0, true);
  }
  mfrc522.PICC_ReadCardSerial();
  printLcd(F("Tarjeta encontrada"), 0, 0, true);
  //delay(3000);
}


bool autenticar(char Key) {
  /*
     esta funcion autentica las tarjetas para poder trabajar con ellas
  */
  if (!(Key == 'a' || Key == 'A' || Key == 'b' || Key == 'B')) {
    Serial.println(F("ERROR: Argumento incorrecto"));//Borrar despues de
        fase debug
    return false;
  }
  const byte trailerBlock = 11;
  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  if (Key == 'a' || Key == 'A') {
    // Authenticate using key A
    Serial.println(F("Authenticating using key A..."));
    printLcd(F("Autenticando key A"), 0, 0, 1);
    status = (MFRC522::StatusCode)
        mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
        trailerBlock, &newKey, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
      printLcd(F("Error Autenticando"), 0, 0, 1);
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return false;
    } else {
      Serial.print(F("PCD_Authenticate() succes: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return true;
    }

  }
  if (Key == 'b' || Key == 'B') {
    // Authenticate using key B
```

```
    Serial.println(F("Authenticating again using key B..."));
    printLcd(F("Autenticando key B"), 0, 0, 1);
    status = (MFRC522::StatusCode)
        mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B,
        trailerBlock, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
      printLcd(F("Error Autenticando"), 0, 0, 1);
      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return false;
    } else {
      Serial.print(F("PCD_Authenticate() succes: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return true;
    }
  }
}

void serialDump(){
    Serial.println("Dumpeo Serial");
  //************************************************


  if (!checkPicc(false)) // en esta rutina se descarta una tarjeta que no
      pertenezca al sistema.
    return;
    // Show the whole sector as it currently is
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &newKey, 2);
  Serial.println();
}

void writeMaster() {
  Serial.println("Escribir masyer");
  //************************************************


  if (!checkPicc(false)) // en esta rutina se descarta una tarjeta que no
      pertenezca al sistema.
    return;
  //funcion check picc verifica que la tarjeta pertenezca al sistema:
      espera un nuevo tag, autentica y chequea los formatos,
  //ademas tiene un argumento en caso tal que quieras usar un halt interno
      que posee. en este caso no es necesario el halt
```

```
byte sector       = 2;
byte blockAddr    = 8;
byte dataBlock[]  = {
  'j', 'u', 'l', 'i', // 1, 2,  3,  4,
  'a', 'n', 'a', 'y', // 5, 6,  7,  8,
  's', 'i', 'l', 'v', // 9, 10, 255, 12,
  'a', 'n', 'a', 0xff // 13, 14, 15, 16
};
byte trailerBlock = 11;
MFRC522::StatusCode status;
byte buffer[18];
byte size = sizeof(buffer);

// Show the whole sector as it currently is
Serial.println(F("Current data in sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &newKey,
    sector);
Serial.println();

// Read data from the block
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
    &size);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Read() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.print(F("Data in block ")); Serial.print(blockAddr);
    Serial.println(F(":"));
dump_byte_array(buffer, 16); Serial.println();
Serial.println();

// Write data to the block
Serial.print(F("Writing data into block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
dump_byte_array(dataBlock, 16); Serial.println();
status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(blockAddr,
    dataBlock, 16);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Write() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
```

```
  Serial.println();

  // Read data from the block (again, should now be what we have written)
  Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
  Serial.println(F(" ..."));
  status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
      &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
  }
  Serial.print(F("Data in block ")); Serial.print(blockAddr);
      Serial.println(F(":"));
  dump_byte_array(buffer, 16); Serial.println();

  // Check that data in block is what we have written
  // by counting the number of bytes that are equal
  Serial.println(F("Checking result..."));
  byte count = 0;
  for (byte i = 0; i < 16; i++) {
    // Compare buffer (= what we've read) with dataBlock (= what we've
        written)
    if (buffer[i] == dataBlock[i])
      count++;
  }
  Serial.print(F("Number of bytes that match = ")); Serial.println(count);
  if (count == 16) {
    Serial.println(F("Success :-)"));
  } else {
    Serial.println(F("Failure, no match :-("));
    Serial.println(F(" perhaps the write didn't work properly..."));
  }
  Serial.println();

  // Dump the sector data
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
      sector);
  Serial.println();
}

void cleanMaster() {
  Serial.println("Escribir masyer");
  //*************************************************
```

```
if (!checkPicc(false)) // en esta rutina se descarta una tarjeta que no
    pertenezca al sistema.
  return;
//funcion check picc verifica que la tarjeta pertenezca al sistema:
    espera un nuevo tag, autentica y chequea los formatos,
//ademas tiene un argumento en caso tal que quieras usar un halt interno
    que posee. en este caso no es necesario el halt

byte sector       = 2;
byte blockAddr    = 8;
byte dataBlock[]  = {
  'j', 'u', 'l', 'i', // 1, 2,  3,  4,
  'a', 'n', 'a', 'y', // 5, 6,  7,  8,
  's', 0x00, 'l', 'v', // 9, 10, 255, 12,
  'a', 'n', 'a', 0xff // 13, 14, 15, 16
};
byte trailerBlock = 11;
MFRC522::StatusCode status;
byte buffer[18];
byte size = sizeof(buffer);

// Show the whole sector as it currently is
Serial.println(F("Current data in sector:"));
mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &newKey,
    sector);
Serial.println();

// Read data from the block
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
    &size);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Read() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.print(F("Data in block ")); Serial.print(blockAddr);
    Serial.println(F(":"));
dump_byte_array(buffer, 16); Serial.println();
Serial.println();

// Write data to the block
```

```
Serial.print(F("Writing data into block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
dump_byte_array(dataBlock, 16); Serial.println();
status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(blockAddr,
    dataBlock, 16);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Write() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.println();

// Read data from the block (again, should now be what we have written)
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
    &size);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Read() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.print(F("Data in block ")); Serial.print(blockAddr);
    Serial.println(F(":"));
dump_byte_array(buffer, 16); Serial.println();

// Check that data in block is what we have written
// by counting the number of bytes that are equal
Serial.println(F("Checking result..."));
byte count = 0;
for (byte i = 0; i < 16; i++) {
  // Compare buffer (= what we've read) with dataBlock (= what we've
      written)
  if (buffer[i] == dataBlock[i])
    count++;
}
Serial.print(F("Number of bytes that match = ")); Serial.println(count);
if (count == 16) {
  Serial.println(F("Success :-)"));
  printLcd(F("Master borrado"),0,0,1);
} else {
  Serial.println(F("Failure, no match :-("));
  Serial.println(F(" perhaps the write didn't work properly..."));
  printLcd(F("Error borrando master"),0,0,1);
}
Serial.println();
```

```
  // Dump the sector data
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
      sector);
  Serial.println();

}

/*
   Escribe la clave nueva en el tag
*/
void writeNewKey() {
  // In this sample we use the second sector,
  // that is: sector #1, covering block #4 up to and including block #7
  byte sector       = 2;
  byte blockAddr    = 11;
  byte dataBlock[]  = {
    'M', 'a', 'r', 'i', // 1, 2,  3,  4,
    'a', 'n', 0xff, 0x07, // 5, 6,  7,  8,
    0x80, 0x69, 0xff, 0xff, // 9, 10, 255, 12,
    0xff, 0xff, 0xff, 0xff // 13, 14, 15, 16
  };

  byte trailerBlock = 11;
  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);

  // Authenticate using key A
  Serial.println(F("Authenticating using key A..."));
  status = (MFRC522::StatusCode)
      mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
      trailerBlock, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("PCD_Authenticate() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }

  // Show the whole sector as it currently is
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
      sector);
```

```
Serial.println();

// Read data from the block
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
    &size);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Read() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.print(F("Data in block ")); Serial.print(blockAddr);
    Serial.println(F(":"));
dump_byte_array(buffer, 16); Serial.println();
Serial.println();

// Authenticate using key B
Serial.println(F("Authenticating again using key B..."));
status = (MFRC522::StatusCode)
    mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B,
    trailerBlock, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("PCD_Authenticate() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}

// Write data to the block
Serial.print(F("Writing data into block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
dump_byte_array(dataBlock, 16); Serial.println();
status = (MFRC522::StatusCode) mfrc522.MIFARE_Write(blockAddr,
    dataBlock, 16);
if (status != MFRC522::STATUS_OK) {
  Serial.print(F("MIFARE_Write() failed: "));
  Serial.println(mfrc522.GetStatusCodeName(status));
}
Serial.println();

// Read data from the block (again, should now be what we have written)
Serial.print(F("Reading data from block ")); Serial.print(blockAddr);
Serial.println(F(" ..."));
status = (MFRC522::StatusCode) mfrc522.MIFARE_Read(blockAddr, buffer,
    &size);
```

```
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
  }
  Serial.print(F("Data in block ")); Serial.print(blockAddr);
      Serial.println(F(":"));
  dump_byte_array(buffer, 16); Serial.println();

  // Check that data in block is what we have written
  // by counting the number of bytes that are equal
  Serial.println(F("Checking result..."));
  byte count = 0;
  for (byte i = 0; i < 16; i++) {
    // Compare buffer (= what we've read) with dataBlock (= what we've
        written)
    if (buffer[i] == dataBlock[i])
      count++;
  }
  Serial.print(F("Number of bytes that match = ")); Serial.println(count);
  if (count == 16) {
    Serial.println(F("Success :-)"));
  } else {
    Serial.println(F("Failure, no match :-("));
    Serial.println(F(" perhaps the write didn't work properly..."));
  }
  Serial.println();

  // Dump the sector data
  Serial.println(F("Current data in sector:"));
  mfrc522.PICC_DumpMifareClassicSectorToSerial(&(mfrc522.uid), &key,
      sector);
  Serial.println();
}

/*
   rutina que revisa el formato de valor de un tag
*/

bool checkFormatValue(byte blockAddr) {
  /*
    Funcion que revisa si el block tiene formato de Value
  */

  byte buffer[18];
```

```
  byte size = sizeof(buffer);
  MFRC522::StatusCode status;

  Serial.print(F("Reading block ")); Serial.println(blockAddr);
  status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return false;
  }

  if (    (buffer[0] == (byte)~buffer[4])
          &&  (buffer[1] == (byte)~buffer[5])
          &&  (buffer[2] == (byte)~buffer[6])
          &&  (buffer[3] == (byte)~buffer[7])

          &&  (buffer[0] == buffer[8])
          &&  (buffer[1] == buffer[9])
          &&  (buffer[2] == buffer[10])
          &&  (buffer[3] == buffer[11])

          &&  (buffer[12] == (byte)~buffer[13])
          &&  (buffer[12] ==      buffer[14])
          &&  (buffer[12] == (byte)~buffer[15])) {
    Serial.println(F("Block has correct Value Block format."));
    return true;
  }
  else {
    return false;
  }
}

/*
   Ensure that a given block is formatted as a Value Block.
*/
void formatValueBlock(byte blockAddr) {
  byte buffer[18];
  byte size = sizeof(buffer);
  MFRC522::StatusCode status;

  Serial.print(F("Reading block ")); Serial.println(blockAddr);
  status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
```

```
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }

  if (    (buffer[0] == (byte)~buffer[4])
          &&  (buffer[1] == (byte)~buffer[5])
          &&  (buffer[2] == (byte)~buffer[6])
          &&  (buffer[3] == (byte)~buffer[7])

          &&  (buffer[0] == buffer[8])
          &&  (buffer[1] == buffer[9])
          &&  (buffer[2] == buffer[10])
          &&  (buffer[3] == buffer[11])

          &&  (buffer[12] == (byte)~buffer[13])
          &&  (buffer[12] ==      buffer[14])
          &&  (buffer[12] == (byte)~buffer[15])) {
    Serial.println(F("Block has correct Value Block format."));
  }
  else {
    Serial.println(F("Formatting as Value Block..."));
    byte valueBlock[] = {
      0, 0, 0, 0,
      255, 255, 255, 255,
      0, 0, 0, 0,
      blockAddr, ~blockAddr, blockAddr, ~blockAddr
    };
    status = mfrc522.MIFARE_Write(blockAddr, valueBlock, 16);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Write() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
    }
  }
}

int getValue(byte blockUsed) {
  Serial.print("Sacando valor del bloque "); Serial.println(blockUsed);
  long value;
  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  // Show the new value of valueBlockB
  status = mfrc522.MIFARE_GetValue(blockUsed, &value);
  if (status != MFRC522::STATUS_OK) {
```

```
    Serial.print(F("mifare_GetValue() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return -1;
  }
  Serial.print("Valor del bloque "); Serial.println(int(value));
  Serial.println();
  return value;
}

void addValue(long value, byte blockUsed) {

  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  if (value >= 0) {
    /*
       Si value es mayor que 0 por supuesto que se va a sumar, pero si es
          menor
       simplemente se utilizara un decrement para que se reste del valor
          total
       almacenado en el tag.
    */
    Serial.print("Sumando "); Serial.print(value); Serial.print(" al
        bloque "); Serial.println(blockUsed);
    status = mfrc522.MIFARE_Increment(blockUsed, value);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Increment() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      printLcd(F("Error tiempo"), 0, 0, 1);
      printLcd(F("Incrementando"), 0, 1, 0);
      return;
    }
  } else {
    Serial.print("Restando "); Serial.print(value); Serial.print(" al
        bloque "); Serial.println(blockUsed);
    status = mfrc522.MIFARE_Decrement(blockUsed, -value);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Decrement() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
    }
  }
  status = mfrc522.MIFARE_Transfer(blockUsed);
  if (status != MFRC522::STATUS_OK) {
```

```
    Serial.print(F("MIFARE_Transfer() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
}

void emptyBlock() {

  Serial.println("Timeblock=\n");
  int delta = getValue(timeBlock);
  addValue(-delta, timeBlock);
  delta = getValue(timeBlock);
  Serial.print("delta = "); Serial.println(delta);
  Serial.println("Serviceblock=\n");
  delta = getValue(serviceBlock);
  addValue(-delta, serviceBlock);
  delta = getValue(serviceBlock);
  Serial.print("delta = "); Serial.println(delta);
}
```

5.1.2 Controlador de válvulas

VALVES_CONTROLLER.ino

```
//Inicializar display de 7 segmentos

byte dataPins[] = {6, 5, 4, 3};
byte DigitOrder[] = {14, 15, 16, 17};

//Inicializar RC522

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN      9          // Configurable, see typical pin layout
    above
#define SS_PIN       10         // Configurable, see typical pin layout
    above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

//Definir objetos key para usarlos para autenticar los sectores de los
    tags RF

MFRC522::MIFARE_Key key;   // Clave por defecto en los tags
MFRC522::MIFARE_Key newKey; //Clave usada en el sistema


const byte timeBlock = 9;    // bloque donde se guarda el tiempo para que
    la maquina funcione
const byte serviceBlock = 10; // bloque donde se guarda la eleccion de
    servicios

const byte BUZZER_PIN = 2;     // pin que hace sonar el buzzer
const byte valves = 7;         //Pin donde estara conectada la
    electrovalvula del aire
const byte gateControl = 18;   // pin donde esta conectada la
    electrovalvula del agua

unsigned long start;           // Variable que marca el inicio del proceso

void setup() {
  inicializar_7segmentos();       // Rutina para inicializar los 7
      segmentos
```

```
  inicializarRF522();             //Rutina para inicializar RC522
  INICIALIZAR_PINES_DE_VALVULAS(); // RUTINA PARA INICIALIZAR PINES DE
      VALVULAS

}

void loop() {
  /*Disp7(0,0);
    Disp7(1,1);
    Disp7(2,2);
    Disp7(7,3);
    delay(1000);*/
  BUZZER_RING(1);
  APAGA_DIGITOS();
  if (!checkPicc(false)) {
    // RUTINA QUE ESPERA POR UNA NUEVA TARJETA Y LA VERIFICA, SI NO
        PERTENECE AL SISTEMA
    // LA DESCARTA, DETIENE EL FUNCIONAMIENTO DE ESA TARJETA Y VUELVE A
        ESPERAR OTRA TARJETA.
    Serial.println("Tarjeta no pertenece al sistema");
    //Disp7(9, 0);
    RFID_HALT();
    BUZZER_ERROR();
    delay(1000);

    APAGAR_TODO();
    return;
  }
  //Disp7(7, 0);
  delay(1000);
  int services = getValue(serviceBlock);//TOMA LOS VALORES DE SERVICIO EN
      EL TAG

  long timePaid = getValue(timeBlock);//TOMA LOS VALORES DE TIEMPO EN EL
      TAG

  timePaid = timePaid * 60000; // EL VALOR DE MINUTOS LO MULTIPLICA POR
      1000 PORQUE
  // AL USAR LA FUNCION MILLIS() TODOS LOS VALORES ESTARAN
  // EN MILISEGUNDOS, Y LO MULTIPLICA POR 60 PORQUE ES
  // NECESARIO PASAR LOS VALORES DE SEGUNDOS A MINUTOS

  if (timePaid > 0)// SI EL TAG TIENE TIEMPO LO BORRA
    emptyBlock();
```

```
  Serial.print("Hay "); Serial.print(timePaid); Serial.println(" tiempo
      pago en la tarjeta");

  RFID_HALT();//RUTINA DE HALT DE UNA TARJETA

  Serial.print("Hay "); Serial.print(timePaid); Serial.println(" tiempo
      pago en la tarjeta");

  VALVES_CONTROL(services, timePaid); //FUNCION QUE ENCIENDE LAS VALVULAS

  delay(1000);


  // LAS SIGUIENTES DOS LINEAS REVISAN SI HAY UNA NUEVA TARJETA
  // (O LA MISMA CON MAS TIEMPO)
  // Y VUELVE A CHECKEARLA, SI TIENE MAS TIEMPO EMPIEZA DESDE CERO
  // SIN HABER APAGADO LAS VALVULAS O EL SISTEMA.
  if (mfrc522.PICC_IsNewCardPresent()) {
    Serial.println("Nueva tarjeta detectada");
    return;
  }

  APAGAR_TODO(); // SI NO HAY UNA NUEVA (O LA MISMA CON MAS TIEMPO)
  //EL SISTEMA APAGA TODO Y VUELVE A EMPEZAR
}
```

FUNCIONES_RFID.ino

```
/**
   ----------------------------------------------------------------------------
   This is a MFRC522 library example; see
      https://github.com/miguelbalboa/rfid
   for further details and other examples.

   NOTE: The library file MFRC522.h has a lot of useful info. Please read
      it.

   Released into the public domain.
   ----------------------------------------------------------------------------
   This sample shows how to read and write data blocks on a MIFARE Classic
      PICC
   (= card/tag).

   BEWARE: Data will be written to the PICC, in sector #1 (blocks #4 to
      #7).


   Typical pin layout used:
   -----------------------------------------------------------------------------------------
               MFRC522     Arduino      Arduino   Arduino   Arduino
                  Arduino
               Reader/PCD  Uno          Mega      Nano v3   Leonardo/Micro
                  Pro Micro
   Signal      Pin         Pin          Pin       Pin       Pin             Pin
   -----------------------------------------------------------------------------------------
   RST/Reset   RST         9            5         D9        RESET/ICSP-5    RST
   SPI SS      SDA(SS)     10           53        D10       10              10
   SPI MOSI    MOSI        11 / ICSP-4  51        D11       ICSP-4          16
   SPI MISO    MISO        12 / ICSP-1  50        D12       ICSP-1          14
   SPI SCK     SCK         13 / ICSP-3  52        D13       ICSP-3          15

*/



byte dataMaster[]  = {
  'j', 'u', 'l', 'i', // 1, 2,  3,  4,
  'a', 'n', 'a', 'y', // 5, 6,  7,  8,
  's', 'i', 'l', 'v', // 9, 10, 255, 12,
  'a', 'n', 'a', 0xff // 13, 14, 15, 16
```

```
};

void RFID_HALT() {
  // Halt PICC
  mfrc522.PICC_HaltA();
  // Stop encryption on PCD
  mfrc522.PCD_StopCrypto1();
}
void inicializarRF522() {

  Serial.begin(9600); // Initialize serial communications with the PC
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  byte help1[] = {'M', 'a', 'r', 'i', 'a', 'n', 'a'}; // Clave de sistema
  byte help[] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07}; // Clave de
      sistema
  // Prepare the key (used both as key A and as key B)
  // using FFFFFFFFFFFFh which is the default at chip delivery from the
      factory
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xff; //Definir los bytes de la clave por defecto
    newKey.keyByte[i] = help1[i]; // Definir los bytes de la clave usada
        por el sistema
  }

  Serial.println(F("Scan a MIFARE Classic PICC to demonstrate read and
      write."));
  Serial.print(F("Using key (for A and B):"));
  dump_byte_array(key.keyByte, MFRC522::MF_KEY_SIZE);
  dump_byte_array(newKey.keyByte, MFRC522::MF_KEY_SIZE);
  Serial.println();

  Serial.println(F("BEWARE: Data will be written to the PICC, in sector
      #2"));
}
/**
   Helper routine to dump a byte array as hex values to Serial.
*/
void dump_byte_array(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}
```

```
void waitForPICC() {
  /*
     Esta funcion espera por una nueva tarjeta para empezar a trabajar
  */
  Serial.println("Coloque nueva tarjeta");
  while (! mfrc522.PICC_IsNewCardPresent()) {
    Serial.println("Tarjeta");
  }
  mfrc522.PICC_ReadCardSerial();
}


bool autenticar(char Key) {
  /*
     esta funcion autentica las tarjetas para poder trabajar con ellas
  */
  if (!(Key == 'a' || Key == 'A' || Key == 'b' || Key == 'B')) {
    Serial.println(F("ERROR: Argumento incorrecto"));//Borrar despues de
        fase debug
    return false;
  }
  const byte trailerBlock = 11;
  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  if (Key == 'a' || Key == 'A') {
    // Authenticate using key A
    Serial.println(F("Authenticating using key A..."));

    status = (MFRC522::StatusCode)
        mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
        trailerBlock, &newKey, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {

      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return false;
    } else {
      Serial.print(F("PCD_Authenticate() succes: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return true;
    }
```

```
  }
  if (Key == 'b' || Key == 'B') {
    // Authenticate using key B
    Serial.println(F("Authenticating again using key B..."));

    status = (MFRC522::StatusCode)
        mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_B,
        trailerBlock, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {

      Serial.print(F("PCD_Authenticate() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return false;
    } else {
      Serial.print(F("PCD_Authenticate() succes: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return true;
    }
  }
}


bool checkPicc(bool halt) {
  waitForPICC();//Rutina que espera una tarjeta en el lector
  if (autenticar('a') && autenticar('b')) //autentica con las claves
      elegidas en los argumentos
  {

  } else {
    return false;
  }
  if (checkFormatValue(timeBlock) && checkFormatValue(serviceBlock))
      //autentica con las claves elegidas en los argumentos
  {

  } else {
    return false;
  }
  if (!halt)
    return true;
  // Halt PICC
  mfrc522.PICC_HaltA();
  // Stop encryption on PCD
```

```
  mfrc522.PCD_StopCrypto1();
  return true;

}

/*
   rutina que revisa el formato de valor de un tag
*/

bool checkFormatValue(byte blockAddr) {
  /*
    Funcion que revisa si el block tiene formato de Value
  */

  byte buffer[18];
  byte size = sizeof(buffer);
  MFRC522::StatusCode status;

  Serial.print(F("Reading block ")); Serial.println(blockAddr);
  status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return false;
  }

  if (    (buffer[0] == (byte)~buffer[4])
          &&  (buffer[1] == (byte)~buffer[5])
          &&  (buffer[2] == (byte)~buffer[6])
          &&  (buffer[3] == (byte)~buffer[7])

          &&  (buffer[0] == buffer[8])
          &&  (buffer[1] == buffer[9])
          &&  (buffer[2] == buffer[10])
          &&  (buffer[3] == buffer[11])

          &&  (buffer[12] == (byte)~buffer[13])
          &&  (buffer[12] ==      buffer[14])
          &&  (buffer[12] == (byte)~buffer[15])) {
    Serial.println(F("Block has correct Value Block format."));
    return true;
  }
  else {
    return false;
```

```
  }
}

/*
   Ensure that a given block is formatted as a Value Block.
*/
void formatValueBlock(byte blockAddr) {
  byte buffer[18];
  byte size = sizeof(buffer);
  MFRC522::StatusCode status;

  Serial.print(F("Reading block ")); Serial.println(blockAddr);
  status = mfrc522.MIFARE_Read(blockAddr, buffer, &size);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Read() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }

  if (    (buffer[0] == (byte)~buffer[4])
          &&  (buffer[1] == (byte)~buffer[5])
          &&  (buffer[2] == (byte)~buffer[6])
          &&  (buffer[3] == (byte)~buffer[7])

          &&  (buffer[0] == buffer[8])
          &&  (buffer[1] == buffer[9])
          &&  (buffer[2] == buffer[10])
          &&  (buffer[3] == buffer[11])

          &&  (buffer[12] == (byte)~buffer[13])
          &&  (buffer[12] ==      buffer[14])
          &&  (buffer[12] == (byte)~buffer[15])) {
    Serial.println(F("Block has correct Value Block format."));
  }
  else {
    Serial.println(F("Formatting as Value Block..."));
    byte valueBlock[] = {
      0, 0, 0, 0,
      255, 255, 255, 255,
      0, 0, 0, 0,
      blockAddr, ~blockAddr, blockAddr, ~blockAddr
    };
    status = mfrc522.MIFARE_Write(blockAddr, valueBlock, 16);
    if (status != MFRC522::STATUS_OK) {
```

```
      Serial.print(F("MIFARE_Write() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
    }
  }
}

long getValue(byte blockUsed) {
  Serial.print("Sacando valor del bloque "); Serial.println(blockUsed);
  long value;
  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  // Show the new value of valueBlockB
  status = mfrc522.MIFARE_GetValue(blockUsed, &value);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("mifare_GetValue() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return -1;
  }
  Serial.print("Valor del bloque "); Serial.println(int(value));
  Serial.println();
  return value;
}

void addValue(long value, byte blockUsed) {

  MFRC522::StatusCode status;
  byte buffer[18];
  byte size = sizeof(buffer);
  if (value >= 0) {
    /*
       Si value es mayor que 0 por supuesto que se va a sumar, pero si es
           menor
       simplemente se utilizara un decrement para que se reste del valor
           total
       almacenado en el tag.
    */
    Serial.print("Sumando "); Serial.print(value); Serial.print(" al
        bloque "); Serial.println(blockUsed);
    status = mfrc522.MIFARE_Increment(blockUsed, value);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Increment() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
```

```
    }
  } else {
    Serial.print("Restando "); Serial.print(value); Serial.print(" al
        bloque "); Serial.println(blockUsed);
    status = mfrc522.MIFARE_Decrement(blockUsed, -value);
    if (status != MFRC522::STATUS_OK) {
      Serial.print(F("MIFARE_Decrement() failed: "));
      Serial.println(mfrc522.GetStatusCodeName(status));
      return;
    }
  }
  status = mfrc522.MIFARE_Transfer(blockUsed);
  if (status != MFRC522::STATUS_OK) {
    Serial.print(F("MIFARE_Transfer() failed: "));
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
}

void emptyBlock() {

  Serial.println("Timeblock=\n");
  int delta = getValue(timeBlock);
  addValue(-delta, timeBlock);
  delta = getValue(timeBlock);
  Serial.print("delta = "); Serial.println(delta);
  Serial.println("Serviceblock=\n");
  delta = getValue(serviceBlock);
  addValue(-delta, serviceBlock);
  delta = getValue(serviceBlock);
  Serial.print("delta = "); Serial.println(delta);
}
```

FUNCIONES_VALVES.ino

```
void VALVES_CONTROL(int services, long timePaid) {
  /*
     ESTA FUNCION ENCIENDE LAS VALVULAS Y LA LUZ DE LA SILLA POR UN CIERTO
        TIEMPO
  */
  //bool once = true;
  switch (services) {
    case 10:
      Serial.println("Utilizaras solo luz");
      // solo luz
      start = millis();

      while (millis() - start < timePaid) {
        int CURRENT_TIME = (timePaid / 1000 - ((millis() - start) / 1000))
            / 60;
        Serial.println(CURRENT_TIME);
        DISPLAY_TIME(CURRENT_TIME);
        if (CURRENT_TIME == 5) {
          BUZZER_RING(3);
          delay(57000);

        }
        ENCENDER_SOLO_LUZ();
      }
      break;
    case 20:
      Serial.println("Utilizaras todo");

      start = millis();
      while (millis() - start < timePaid) {
        int CURRENT_TIME = (timePaid / 1000 - ((millis() - start) / 1000))
            / 60;
        Serial.println(CURRENT_TIME);
        DISPLAY_TIME(CURRENT_TIME);
        if (CURRENT_TIME == 5) {
          BUZZER_RING(3);
          delay(57000);
        }
        ENCENDER_TODO();
      }
      break;
  }
```

```
  BUZZER_RING(2);
  APAGA_DIGITOS();
}

void BUZZER_RING(int m) {
  for (int i = 0; i < m; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(500);
    digitalWrite(BUZZER_PIN, LOW);
    delay(500);
  }
  return;
}

void BUZZER_ERROR() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(200);
    digitalWrite(BUZZER_PIN, LOW);
    delay(200);
  }
  return;
}
void INICIALIZAR_PINES_DE_VALVULAS() {
  pinMode(valves, OUTPUT);
  pinMode(gateControl, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);     //inicializar buzzer
  APAGAR_TODO();
}

void ENCENDER_TODO() {
  //bool on = true;
  digitalWrite(gateControl, LOW); //encender el optoacoplador
  digitalWrite(valves, HIGH);    //Encender valvulas
}
void ENCENDER_SOLO_LUZ() {
  //bool on = false;
  digitalWrite(valves, LOW);     //apagar valvulas
  digitalWrite(gateControl, LOW); //Encender optoacoplador
}
void APAGAR_TODO() {
  //bool on = false;
  Serial.println(F("Apagar todo!"));
```

```
  digitalWrite(gateControl, HIGH);
  digitalWrite(valves, LOW);
}
```

FUNCIONES_7SEGMENTOS.ino

```
void inicializar_7segmentos() {
  for (int i = 0; i < 4; i++) {
    pinMode(dataPins[i], OUTPUT);
    pinMode(DigitOrder[i], OUTPUT);
  }
  for (int i = 0; i < 4; i++) {
    digitalWrite(DigitOrder[i], HIGH);
  }
}

byte Digit[10][4] = // Arduino UNO va muy justo de memoria. Por eso lo
{ // definimos como byte y no como int
  { 0, 0, 0, 0 },  // 0
  { 0, 0, 0, 1 },  // 1
  { 0, 0, 1, 0 },  // 2
  { 0, 0, 1, 1 },  // 3
  { 0, 1, 0, 0 },  // 4
  { 0, 1, 0, 1 },  // 5
  { 0, 1, 1, 0 },  // 6
  { 0, 1, 1, 1 },  // 7
  { 1, 0, 0, 0 },  // 8
  { 1, 0, 0, 1 }   // 9
};

void showThemAll(int N) {
  for (int i = 0; i < 4; i++) {
    digitalWrite(dataPins[i], Digit[N][i]);
    digitalWrite(DigitOrder[i], LOW);
  }

}

void APAGA_DIGITOS() {
  for (int i = 0; i < 4; i++) {
    digitalWrite(DigitOrder[i], HIGH);
  }
}

void Disp7(int N, int led) {
  APAGA_DIGITOS();

  for (int i = 0; i < 4; i++) {
```

```
    digitalWrite(dataPins[i], Digit[N][i]);
  }

  digitalWrite(DigitOrder[led], LOW);
  delay(5);
}
void DISPLAY_TIME(int timeUsed) {

  if (timeUsed < 60) {
    Disp7(0, 0);
    Disp7(0, 1);
    int minutes = timeUsed % 60;
    int minute1 = minutes / 10;

    Disp7(minute1, 2);
    int minute2 = minutes % 10;

    Disp7(minute2, 3);
    return;
  }
  else {
    int hour1 = timeUsed / 600;
    Disp7(hour1, 0);

    int hour2 = timeUsed / 60 ;
    Disp7(hour2, 1);

    int minutes = timeUsed % 60;
    int minute1 = minutes / 10;
    Disp7(minute1, 2);

    int minute2 = minutes % 10;
    Disp7(minute2, 3);

    return;
  }
}
```

Bibliografía

[1] Arduino. Arduino - home, 2017.

[2] Arduino. Using an arduino as an avr isp (in-system programmer), 2017.

[3] Atmel. Microcontrollers (mcus), 2016.

[4] Capio. Las ventajas de la contabilidad de costes en tu empresa, 2016.

[5] D. Gadre. *Programming and Customizing the AVR Microcontroller.* McGraw Hill, Columbus, OH 43218, USA, 2000.

[6] T. Igoe, D. Coleman, and B. Jepson. *Beginning NFC: Near Field Communication with Arduino, Android, and PhoneGap.* O'Reilly Media, Inc., 1005 Gravenstein Hwy N Sebastopol, CA 95472 California., 2014.

[7] M. Margolis. *Arduino Cookbok.* O'Reilly Media, 1005 Gravenstein Hwy N Sebastopol, CA 95472 California., 2011.

[8] Miguel Balboa. Mfrc522 library for arduino, 2012.

[9] NXP Semiconductors. *Standard performance MIFARE and NTAG frontend*, 4 2016. Rev. 3.9.

[10] Phillips Semiconductors. *TRIACS BT137 SERIES*, 10 1997. Rev. 3.9.

[11] Portafolio. La robótica en las empresas no es cosa de ciencia ficción. *Portafolio*, 25 y 26 de Marzo:pag. 6, 2017.

[12] V. K. Vaishnavi and W. Kuechler. *Desing Science Research Methods and Paterns: Innovating Information and Communication Technology.* CRC Press, Boca Raton, 2015.